University of California
Santa Cruz

COLLISION AVOIDANCE AND RESOLUTION MULTIPLE ACCESS

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Engineering

by

Rodrigo Garcés

March 1999

The dissertation of Rodrigo Garcés is
approved:

JJ Garcia-Luna-Aceves

Glen Langdon

Darrell Long

Dean of Graduate Studies and Research

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAR 1999**	2. REPORT TYPE	3. DATES COVERED **00-03-1999 to 00-03-1999**

4. TITLE AND SUBTITLE **Collision Avoidance and Resolution Multiple Access**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**175**	

Contents

List of Figures

List of Tables

COLLISION AVOIDANCE AND RESOLUTION MULTIPLE ACCESS

Rodrigo Garcés

ABSTRACT

Multiple-access interference constitutes a major cause of throughput degradation in wireless networks. The focus of this thesis is the design and analysis of medium access control protocols that mitigate multiple access interference by resolving collisions of small control packets used to avoid the collision of much larger data packets. An upper bound on the average cost of resolving collisions using a deterministic tree-splitting algorithm is derived. This bound is then applied to the compute the average channel utilization in a fully connected network with a large number of stations. Under light-load conditions, collision avoidance and resolution (CARMA) protocols achieve the same average throughput as floor acquisition multiple access (FAMA) protocols. It is also shown that, as the arrival rate of RTSs increases, the throughput achieved by CARMA protocols is close to the maximum throughput that any FAMA protocol can achieve when propagation delays and the control packets used to acquire the floor are much smaller than the data packet trains sent by stations.

We further introduce the incremental collision resolution multiple access (ICRMA) protocol, which maintains a distributed queue for the transmission of data packets and a stack for the transmission of control packets used in collision resolution. ICRMA dynamically divides the channel into cycles of variable length where each cycle consists of a contention period and a queue-transmission period. The queue-transmission period is a variable-length train of packets that are transmitted by stations that have been added to the distributed transmission queue by successfully completing a collision-resolution round in a previous contention period. During the contention period, stations with one or more packets to send compete for the right to be added to the data-transmission queue using a deterministic tree-splitting algorithm. A single round of collision resolution (i.e., a success, idle or a

collision of control packets) is allowed in each contention period. Simulation and analytical results show that ICRMA's throughput is within 5% of the throughput achieved by the ideal channel access protocol based on a distributed transmission queue and incremental collision resolution.

We also propose a novel "spectrum etiquette" to allow systems from different manufacturers with different physical and medium-access control protocols to co-exist, without monitoring the entire band. This is achieved by means of transmissions over a common, narrow band control channel used to establish collision-free transmission schedules over the channels allocated for data transmission. Because no common physical layer can be assumed among different systems, a control channel is needed in order for the systems to schedule transmissions in the rest of the band. The only means by which systems can communicate with one another over the control channel is the duration of each others' transmissions, which are perceived only as noise. A transmission encoding based on this basic feedback is defined to allow systems to ascertain which system can use which data channel at which time without interference. Analytical and simulation results are presented showing that the proposed etiquette is fair to all the co-existing systems and fully utilizes the spectrum. It also provides bounded delays for data-channel acquisition time by any given system and provides minimum channel-use guarantees.

Acknowledgments

First and foremost, I would like to thank my advisor, Professor J. J. Garcia-Luna-Aceves for being my advisor and for teaching us all to have fun while doing good research. He has been the best possible advisor, both in terms of research and in terms of professional advice.

I thank Professors Darrell Long and Glen Langdon for serving on my dissertation committee. I am grateful to Professor Raphael Rom for working with me on the spectrum etiquette project.

I thank Carol Mullane for her friendly attitude, approachability and for guiding me through the many administrative paperwork and deadlines. As to the "coconauts" I thank them for their friendship and I wish the best of lucks in the future. I am also grateful to the Greek gang (Christos, Diamantis, Dimitrios, and Lampros), to Gil and Alice for their friendship. I dedicate this dissertation to each member of my family.

The text of this dissertation includes material that has been previously published in [18, 19, 20, 21, 22, 23] and [24]. The co-authors listed in these publications directed and supervised the research which forms the basis of this dissertation.

The research reported in this dissertation was supported in part by the Defense Advance Research Projects Agency (DARPA) under grants DAAB07-95-C-D157 and DAAH04-96-10210, by the University of California under a MICRO grant with SUN Microsystems, and by a grant from Ray Theon.

1. Introduction

1.1 Background and Motivation

Interference is inherent in all wireless systems and is one of the most important issues to be addressed in the design, operation, and maintenance of wireless communication systems. There are several different types of interference: adjacent-channel interference, foreign interference, busy interference, co-channel interference, and contention interference.

Adjacent-channel interference occurs due to equipment limitations, such as frequency instability, receiver bandwidth, and filtering. Moreover, because channels are kept very close to each other for maximum spectrum efficiency, the random fluctuation of the signal due to fading and near-far-effects, aggravates this problem. There are several efficient strategies to mitigate the effects of adjacent-channel interference. Better filters can be constructed. The total frequency spectrum can be split into two half, one for the up-link and the other half for the down-link. Adjacent-channel interference can also be minimized by avoiding the use of adjacent channels.

Foreign interference are caused by noise sources that are not within the network. The noise can take on any interference pattern, ranging from short bursts lasting only microseconds to long bursts lasting hundreds of milliseconds, to continual interference corrupting channels for long time periods. There is not much that can be done to avoid foreign interference other than avoiding the channels that are affected.

Co-channel interference occurs when two or more independent signals are transmitted simultaneously in the same frequency band. The communication between one pair of stations affects the packet transmission between other pairs. It arises because the same frequencies are reused many times. There are several ways to mitigate co-channel interference. Directional antennas will decrease the effects of this form of interference. Reducing the modulation rate, increasing forward error correction (FEC), data-link fragmentation or power control schemes make the system less sensitive to this form of interference.

Busy interference occurs when a station targets a receiver that is already busy transmitting to another station. Busy interference is very common in half-duplex systems. Finally, contention interference occurs when two or more stations target the same receiver.

When independent devices share the same medium, as it is the case in multiple access channels, the main problem consists in allocating the channel capacity among the various users and minimizing the effects of interference. The layer that controls the access to the physical layer in multiple access channels is called the *multiple access control* (MAC) layer. An efficient MAC protocol must limit the number of errors caused by the collision of simultaneous transmissions, limit the amount of time the transmission channel is idle, and increase the amount of time the channel is used to transmit data.

Current network protocol architectures use a layering approach, which allows for a hierarchical modularity in their designs (e.g., OSI, ARPANET, DECNET). The MAC layer is responsible for providing channel access for outgoing packets from the higher layers to the physical layer by allocating the medium among all of the devices, and for passing packets received by the physical layer up the higher layers. The MAC layer lies between the logical link control (LLC) layer and the physical layer.

MAC protocols can be classified into two main types, conflict-free, and contention-based schemes. Conflict-free protocols ensure that a transmission will not be interfered by another transmission. This can be done by allocating the channel to the users either statically (e.g., TDMA [48], FDMA and CDMA [43]), or dynamically (e.g., BRAM [10], MSAP [34]). On the other hand, in contention-based schemes, stations content for the channel on a packet by packet basis (e.g., ALOHA [1] and CSMA [33]).

Several MAC protocols have been proposed over the past few years that are based on three- or four-way handshake procedures meant to reduce the number of collisions among data packets, thereby providing better performance than the basic ALOHA or CSMA protocols [4, 5, 7, 16, 30, 36, 50]. The concept of "floor acquisition" was first introduced by Fullmer and Garcia-Luna-Aceves [16] for MAC protocols based on such handshake procedures. In a single-channel network, floor acquisition entails allowing one and only

one station at a time to send data packets without collisions. To achieve this, a station that wishes to send one or multiple data packets must send a request-to-send packet (RTS) to an intended destination and receive a clear-to-send packet (CTS) from it, before it is allowed to transmit any data. RTSs are required to last a minimum amount of time that is a function of the channel propagation time. Although floor acquisition multiple access (FAMA) protocols are able to sustain higher loads than CSMA [16], their throughput still degrades rapidly once stations start retransmitting unsuccessful RTSs that collide repeatedly with other RTSs. Eventually FAMA protocols become unstable.

Several approaches have been proposed in the past combining contention schemes with reservations, token passing, or polling (e.g., [31, 34, 51, 45, 48]). A drawback of fixed assignments protocols is that the channel is wasted if the stations do not use their allocated frames. Establishing and modifying time or frequency assignments is complex. On the other hand, polling requires a central station to poll the packet radios, and is efficient only if the majority of the packet radios are busy transmitting.

A way to stabilize the system is by increasing the retransmission delays. However, a more efficient way can be devised by using collision resolution. Several stable MAC protocols have been proposed in the past based on tree-splitting algorithms for collision resolution (e.g., [9, 17, 40]). In these protocols data packets are used to resolve collisions achieving throughputs below 0.6 [55] for a single channel and fully connected networks.

The focus of this dissertation is the design and analysis of MAC protocols that mitigate multiple access interference by means of two basic mechanisms: (a) use of small control packets requesting or reserving the right to access the channel, and (b) a method of resolving collisions among such control packets. We introduce and analyze a family of protocols based on collision avoidance and resolution.

In contrast to prior MAC protocols based on collision resolution, the protocols introduced in this dissertation offer very high throughput at heavy loads and do not require the use of time slotting. We apply these protocols to multi-hop packet radio networks and to networks in which not all nodes share a common physical layer, which is the assumption in

all MAC protocols.

This dissertation advances the state of the art in several ways, ranging from a better understanding of the performance of collision resolution algorithms to the development of new MAC protocols that are far more efficient than MAC protocols previously proposed, or which permit the coexistence of heterogeneous systems.

1.2 Structure of the Dissertation

This dissertation is organized in chapters that present different aspects of MAC protocols based on collision resolution and avoidance.

Chapter 2 describes the deterministic-tree-splitting algorithm to resolve collisions as soon as they occur. The deterministic-tree-splitting algorithm and floor acquisition are the core for all the protocols presented in this dissertation. We present a detailed analysis for the average number of collision resolution steps required by the algorithm as well as upper bounds.

Chapter 3 presents and analyzes the throughput achieved by the deterministic tree-splitting algorithm assuming a slotted channel. We first compare our results with the slotted ALOHA results presented by Roberts [44]. We then review the probabilistic tree-splitting algorithm proposed independently by Capetanakis [9], Tsybakov and Mikhailovic [54], and Hayes [27]. We show that the average number of steps required by the probabilistic tree-splitting collision resolution protocol is greater than the average number of steps required by the deterministic version of the tree-splitting protocol. The analysis shows that if the number of nodes in the deterministic model goes to infinity, the model converges to the probabilistic model, i.e., the probabilistic model is an upper bound for the deterministic model. Later in the chapter we compare the deterministic tree-splitting algorithm to DQRAP [57], which is a competitive protocol based on collision resolution. Once again, our analysis shows that the deterministic tree-splitting algorithm outperforms DQRAP.

Even though the deterministic tree-splitting algorithm outperformed each of this algorithms, the throughput results left much to be desired. In Chapter 4 we introduce the

concept of carrier sensing and floor acquisition combining this concepts into the first member of the CARMA family of protocols. The goal is to develop a MAC protocol that fully resolves collisions while having an acceptable throughput. CARMA (for collision avoidance and resolution multiple access) uses non-persistent carrier sensing for the transmission of RTSs and a tree-splitting algorithm to resolve collisions of RTSs. The basic deterministic tree-splitting algorithm used in CARMA is similar to the methods used in other protocols based on the identities of nodes.

In the past, several MAC protocols have been proposed based on collision resolution (e.g., RAMA [2], TRAMA [31], DQRUMA [32],DQRAP [57]). CARMA is a stable and very efficient protocol that does not require time slotting or availability of base stations capable of detecting multiple simultaneous transmissions.

Recently MAC protocols have been proposed that build a separate queue for the transmission of data packets, in addition to the stack or queue of the control packets used for collision resolution. However, prior protocols based on data transmission queues and collision resolution require time slots or mini-slots for the transmission of control packets [53, 57].

Chapter 5 introduces a stable multiple access protocol for broadcast channels shared by bursty stations, which we call CARMA-NTQ (for collision avoidance and resolution multiple access with non-persistence and transmission queues). CARMA-NTQ provides dynamic reservations of the channel, together with collision resolution of the reservations requests based on the deterministic tree-splitting algorithm introduced in Chapter 2. CARMA-NTQ builds a dynamically-sized cycle that grows and shrinks depending upon traffic demand. Each cycle consists of a contention period and a queue-transmission period during which one or more stations transmit data packets without collision. A position in the transmission queue is allocated to an individual station during the contention period, and a station can continue to transmit in this position as long as it has data to send to any other station in the network. Stations compete to acquire the right to be in the transmission queue based on the deterministic tree-splitting algorithm.

Chapter 6 introduces a new stable multiple access protocol for broadcast channels shared by multiple stations, which we call the incremental collision resolution multiple access (ICRMA) protocol. ICRMA is an improvement over CARMA-NTQ in two aspects; access delay and throughput. ICRMA decreases the time for a station to be added to the transmission queue since the collision resolution is done on a continuous basis, one step per contention period. Access time to the channel is divided into rounds of transmissions for all members of the transmission queue, which we call a queue-transmission period, followed by short contention periods during which stations attempt to join the queue. The queue-transmission period is a variable-length train of packets from stations that have been added to the transmission queue by successfully completing a collision-resolution round in a previous contention period. A single round of collision resolution (i.e., a success, idle or a collision of control packets) is allowed in each contention period.

Chapter 7 introduces a new stable receiver-initiated, multi-hop, multichannel, multiple access protocol with collision resolution, which we call CARMA-MC (for collision avoidance and resolution multiple access multi channel). CARMA-MC is a receiver initiated protocol, which dynamically divides the channel into cycles of variable length; each cycle consists of a receiving period and a transmission period. The transmission period has a variable-length duration. During the receiving period, stations with one or more packets to send compete for the right to acquire the floor using a deterministic tree-splitting algorithm.

Chapter 8 proposes a listen-before-transmit etiquette based on power sensing over a control channel used to schedule access to the rest of the band, which is partitioned into data channels. The etiquette is intended for heterogeneous systems sharing a common broad band. Each system consists of any set of nodes using the same PHY and MAC protocols, and two nodes from different systems cannot decode one another's transmissions in any data channel of the band. Each data channel is meant to be used by an individual system (i.e., two or more nodes using the same PHY and MAC layers) on long-term and persistent basis. The control channel is used to exchange information about the band use activity in the area. Prospective transmitters listen to the control channel to get information about

the data channels occupancy. This eliminates the need to listen to the entire wide band.

The only means by which systems can exchange information with one another over the control channel is the duration of each others' transmissions, which are perceived only as noise, and no information is exchanged across systems over the data channels defined in the band. A novel transmission encoding is defined based on this basic control-channel feedback that allows systems to ascertain which system can use which data channel at which time, without interference.

Chapter 9 presents our conclusions and suggestions for future research directions.

2. The Deterministic Tree-Splitting Algorithm

In Chapter 1 we classified MAC protocols into two main types, conflict-free, and contention-based schemes. Conflict-free schemes perform well under heavy load, but waste valuable resources under low load. On the other hand, contention-based schemes perform well under low load but their throughput degrades rapidly once stations start retransmitting unsuccessful packets that collide repeatedly with newly created packets. Eventually, contention-based protocols become unstable. A way to stabilize the system is by increasing the retransmission delays. A more efficient way can be devised by using collision resolution. Several stable MAC protocols have been proposed in the past based on probabilistic tree-splitting algorithms for collision resolution (e.g. [9, 17, 40]). Tree-splitting algorithms are ternary feedback models, i.e. they have three types of steps: *idle, success*, and *collision.*

In this chapter we present and analyze a deterministic version of the tree-splitting algorithm. Our analysis computes the average number of collision-resolution steps required by the algorithm for each of the three types of collision/idle/success steps. Using this analysis we are able to show that the deterministic version requires a smaller number of steps to resolve collisions than the probabilistic tree-splitting algorithm. Our analysis differs from previous ones in the sense that, previous calculations concentrated on the total number of steps and no distinction was ever made between the three types of collision/idle/success steps.

The chapter is organized as follows. The description of the deterministic version of the tree-splitting algorithm is included in Section 2.1. An example describing the algorithm is presented in Section 2.2. The analysis of the algorithm is given in Section 2.3. Finally, in Section 2.4 we derive upper bounds on the average number of collision/idle/success steps.

2.1 Algorithm Description

We begin the description of the deterministic tree-splitting algorithm by assuming that all stations are in line of sight of one another. The channel is slotted and the stations can

transmit a one-slot-size packet only at the beginning of a slot. We assume a *ternary feedback* model, i.e., at the end of each slot the stations in the system know whether the slot was *idle* (no packet was transmitted), *successful* (one packet was transmitted), or there was a *collision* (two or more packets were transmitted). Each station is assigned a unique identifier (ID), a stack, and two variables, $LowID$ and $HiID$. $LowID$ and $HiID$ are respectively the lowest and highest ID numbers of the stations that are initially allowed to transmit. Together, they define the allowed ID interval, $(LowID, HiID)$. If the ID of a station is not within this interval, the station is not allowed to send a packet. New packets that arrive to the system are inhibited from being transmitted while the collision resolution is in progress. Finally, the stack is the storage mechanism for ID intervals waiting to get permission to send a packet.

Collision resolution evolves in terms of collision-resolution steps. There are three types of collision-resolution steps: *collision*, *success*, and *idle*. A POP-stack command is executed each time there is a success or an idle step. A PUSH-stack command is executed each time there is a collision step.

In the first step of a collision-resolution phase all stations in the allowed ID interval try to transmit a packet. If multiple stations transmit a packet causing a *collision step*, then every station in the system divides the allowed ID interval $(LowID, HiID)$ into two ID intervals. The first interval, called the *backoff ID interval*, is $(LowID, \lceil\frac{HiID+LowID}{2}\rceil - 1)$. The second interval, which is the new *allowed ID interval*, is $(\lceil\frac{HiID+LowID}{2}\rceil, HiID)$. Next, each station in the system updates the stack by executing a PUSH-stack command, where the key being pushed is the backoff ID interval. This procedure is repeated each time a collision is detected. If none of the stations within this ID interval transmit a packet an *idle step* occurs. In this case, each station executes a POP-stack command updating the allowed ID interval. The same procedure takes place if, during the collision-resolution step, only one station transmits a packet, i.e., if a *success step* occurs.

The algorithm repeats the above three types of collision-resolution steps, until all the collisions have been resolved. As soon as the backoff stack becomes empty and there are no

values in the allowed ID interval, all stations know that all the collisions have been resolved.

2.2 Example Algorithm's Operations

In the deterministic tree-splitting algorithm each station has a distinct position in the leaves of a binary tree based on its ID. If n is the total number of stations in the system, the binary tree has *2n+1* nodes. The root of the tree is labeled n_r and its right and left child are labeled respectively n_1 and n_0. For the remaining nodes, the labels are defined as follows. If x is the label of node v, then the label of v's left child is x_1 and the label of v's right child is x_0.

For a system with four stations labeled n_{00}, n_{01}, n_{10}, and n_{11}, the corresponding binary tree has seven nodes where the leaves represent the four stations. More precisely, the root of the tree is labeled n_r, its left child is labeled n_1 and is the parent of the leaves labeled n_{11} and n_{10}, while its right child is labeled n_0 and is the parent of the leaves labeled n_{01} and n_{00}. The resulting tree is shown in Fig. 2.1.

Given a tree T, let $T_{<label>}$ be the subtree at node $n_{<label>}$. In our example, the subtree at node n_1 is T_1 while the subtree at node n_0 is T_0.

We now illustrate the deterministic tree-splitting algorithm using the simple example shown in Fig. 2.1. The algorithm evolves in steps. For the sake of simplicity, we have described the evolution of the tree-splitting algorithm up to Step 5. Initially, the backoff stack is empty and the allowed ID interval is (n_{00}, n_{11}).

In Step 1 stations n_{00} and n_{01} each sends a packet, while stations n_{10} and n_{11} are passive. Thus, a collision occurs between n_{00} and n_{01}. All stations detect the beginning of the collision-resolution phase and update their stacks and their $LowID$ as well as their $HiID$ values. Since stations n_{00} and n_{01} are within the backoff ID interval, they are not allowed to send packets until the collisions in the allowed ID interval are resolved.

In the next slot stations n_{10} and n_{11} can send a packet. However, since neither station n_{10} nor station n_{11} wish to send a packet, an idle period occurs (Step 2 in Fig. 2.1). All stations detect that the channel is idle, i.e. that there were no collisions, and they all update

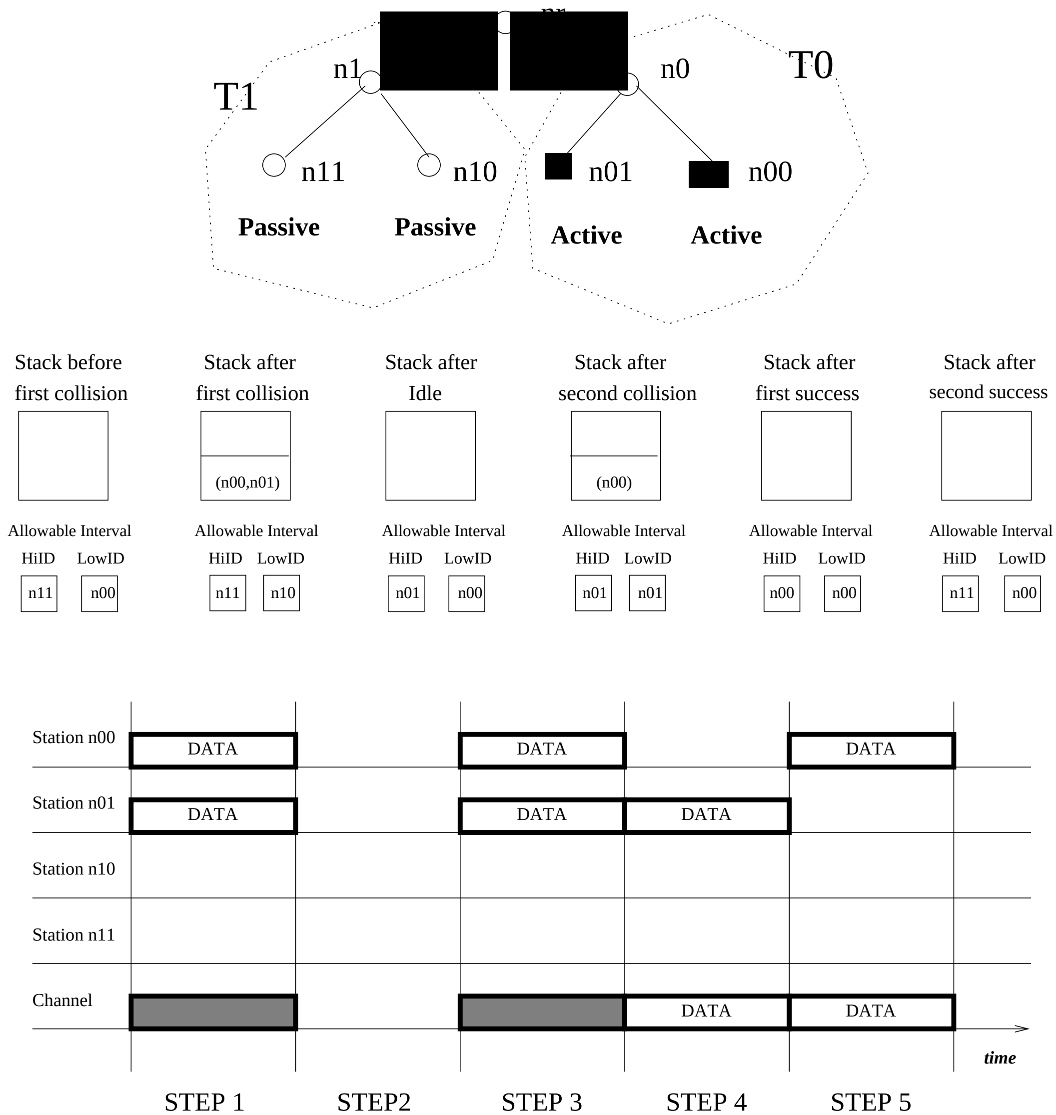

Figure 2.1: Transmission period and tree structure to resolve collisions for a system with $n = 4$ stations out of which $m = 2$ stations have a data packet to send.

their intervals and stacks, executing a POP-stack command. The new allowed interval is (n_{00}, n_{01}).

T_0 can now proceed to resolve its collisions. Both stations n_{00} and n_{01} transmit a packet (Step 3 in Fig. 2.1) and another collision occurs. Since a collision occurred, the allowed ID interval is split, i.e. the subtree T_0 is split into T_{00} and T_{01}. Now, Station n_{01} is within the allowed ID interval while station n_{00} must wait. Since T_{01} has only one station, it transmits

its data packet (Step 4 in Fig. 2.1). A POP-stack command is executed and station n_{00} can now transmit its data packet (Step 5 in Fig. 2.1).

Finally, all stations empty their stacks and update the allowed ID interval permitting all stations to contend in the next collision-resolution phase. Observe that a collision-resolution phase terminates when both the allowed ID interval and the stacks are empty.

2.3 Average Number of Collision Resolution Steps

In this section we compute the average number of idle/success/collision steps required by a collision-resolution algorithm based on a deterministic tree-splitting algorithm to resolve m collisions in a network with n stations, each having a unique ID. We also present upper bounds for the average number of steps.

For the purpose of our analysis we assume that: (a) every station can listen to every other station, (b) the channel introduces no errors, so packet collisions are the only source of errors, (c) stations detect such collisions perfectly, (d) two or more transmissions that overlap in time in the channel must all be re-transmitted, (e) each station can have at most one data packet at a time, and (e) no failures occur.

The binary tree is a structure defined on a finite set of nodes composed of three disjoint sets: a root node, a binary tree called the left subtree and a binary tree called the right subtree. As we have described, there are three possible cases to consider: idle, success, or collision. For each of these cases, we wish to find a recursive equation expressing the average number of steps needed during the collision-resolution phase.

Consider a system with n stations and m packets arriving during a contention time period. Because each station in the system is assigned one or no packets at any given time, a leaf of the binary tree is assigned an "active" or a "passive" label depending on whether or not the station has a data packet to send. Let $\overline{\mathcal{Z}}(n,m)$ denotes the average number of idle steps, $\overline{\mathcal{C}}(n,m)$ denotes the average number of collision steps, and $\overline{\mathcal{S}}(n,m)$ the number of success steps of the collision-resolution algorithm needed to resolve m collisions in a network of n stations. These three functions depend on the number n of leaves and the number m of

stations with packets to send. They compute the average number of steps over all possible permutations of m out of n stations with packets to send.

The number of permutations of a tree with four leaves ($n = 4$), given that two out of the four stations have a data packet ($m = 2$), is six. In our prior example (see Fig. 2.1), stations n_{00} and n_{01} each sends a data packet in the same time slot, while station n_{10} and station n_{11} remain passive. The tree shown in Fig. 2.1 represents just one of the six possible permutations. Let us assign the index i to the ith permutation and compute $\mathcal{S}_i(4,2)$, $\mathcal{Z}_i(4,2)$ and $\mathcal{C}_i(4,2)$ respectively. For our example it is not difficult to see that, the number of success steps is $\mathcal{S}_i(4,2) = 2$, the total number collision steps is $\mathcal{C}_i(4,2) = 2$ and the total number of idle steps is $\mathcal{Z}_i(4,2) = 1$. The same counting procedure can be repeated for each of the $\binom{4}{2} = 6$ permutations of trees with $n = 4$ and $m = 2$ (see Table 2.1). The six possible permutations contribute equally to the total average number of steps $\overline{\mathcal{C}}(4,2)$, $\overline{\mathcal{S}}(4,2)$ and $\overline{\mathcal{Z}}(4,2)$. In general, the average for each of the three types of steps can be calculated by adding each individual permutation cost and dividing by the total number of permutations.

n_{11}	n_{10}	n_{01}	n_{00}	Collision Steps	Success Steps	Idle Steps
passive	passive	active	active	2	2	1
passive	active	passive	active	1	2	0
passive	active	active	passive	1	2	0
active	passive	passive	active	1	2	0
active	passive	active	passive	1	2	0
active	active	passive	passive	2	2	1

Table 2.1: Number of steps required to solve all possible permutations for a tree with $n = 4$ and $m = 2$.

For counting purposes, a subtree that has no packets or only one packet does not have to be explored any further. Counting can stop there and one unit can be added to either $\mathcal{Z}$ or $\mathcal{S}$. Regardless of n, the number of success steps $\mathcal{S}(n,m)$ is always equal to m. Based on our example, the general rules shown in Table 2.2 can be derived.

Rule 1:	$\mathcal{C}$(n,0) = 0	**Rule 4:**	$\mathcal{S}$(n,0) = 0	**Rule 7:**	$\mathcal{Z}$(n,0) = 1
Rule 2:	$\mathcal{C}$(n,1) = 0	**Rule 5:**	$\mathcal{S}$(n,m) = m	**Rule 8:**	$\mathcal{Z}$(n,1) = 0
Rule 3:	$\mathcal{C}$(n,n) = n-1	**Rule 6:**	$\mathcal{S}$(n,n) = n	**Rule 9:**	$\mathcal{Z}$(n,n) = 0

Table 2.2: Rules for the average number of steps.

2.3.1 Average Number of Success Steps

Any tree or subtree, regardless of its size, has a success cost of one step if there exist only one station with a packet to send. This is due to the fact that we only need to visit the root of the tree or the root of the subtree and stop there. In the case of a tree with m stations with packets to send, there are exactly m subtrees with one packet each, otherwise there would be a subtree with no packets or more than two packets creating a collision. In both cases the number of success steps is $\mathcal{S} = 0$. The total average number of success steps for any tree of size n with m stations with a packet to send is $\overline{\mathcal{S}}(n, m) = m$ (see rule 5 in Table 2.2).

2.3.2 Average Number of Collision Steps

To compute the average number of collision steps in a network with n stations and m initial collisions, we need to count the number of collision steps for each of the possible permutations of m collisions among the n leaves of the corresponding binary tree.

Since each tree can be defined in terms of two disjoint binary subtrees and the parent node, the total number of collision steps can be expressed as the number of collision steps of the right subtree, plus the number of collision steps of the left subtree plus one step for the root of the tree. This yields the following equation,

$$\mathcal{C}_{root\ tree} = \mathcal{C}_{right\ subtree} + \mathcal{C}_{left\ subtree} + 1 \tag{2.1}$$

For the example described in Fig. 2.1 and its six permutations shown in Table 2.1, the average number of collision steps $\overline{\mathcal{C}}(4, 2)$ is given by the following recursion.

$$\overline{C}(4,2) = \sum_{i=0}^{2} \frac{\binom{2}{2-i}\binom{2}{i}}{\binom{4}{2}} \left[\overline{C}\,(2, 2-i) + \overline{C}\,(2,i) + 1\right] \tag{2.2}$$

The recursion splits the original tree for $n = 4$ into two subtrees with $n = 2$, where the ratio $\frac{\binom{2}{2-i}\binom{2}{i}}{\binom{4}{2}}$ is the probability of having $2 - i$ stations with a packet in one subtree and i stations with a packet in the other subtree. The following theorem extends this result to the average number of collision steps $\overline{C}(n,m)$ for any $1 < m \leq n$.

Theorem 1: *Let n be the total number of stations in the network and m be the number of stations participating in the tree-splitting algorithm. Then for all $1 < m \leq n$, the average number of collision steps required to resolve the m collisions is*

$$\overline{C}(n,m) = \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{C}\,(\alpha, m-i) + \overline{C}\,(\beta,i) + 1\right] \tag{2.3}$$

where

$$\begin{aligned}
\alpha &= \lceil n/2 \rceil \\
\beta &= n - \alpha = n - \lceil n/2 \rceil \\
\mu &= \begin{cases} 0 & \textit{if } m \leq \alpha \\ m - \alpha & \textit{if } m > \alpha \end{cases} \\
\nu &= \begin{cases} m & \textit{if } m \leq \beta \\ \beta & \textit{if } m > \beta \end{cases}
\end{aligned}$$

Proof: The theorem is proved by induction on n. It is trivial to show that, for all $n \geq 1$, $\overline{C}(n,0) = \overline{C}(n,1) = 0$. When $n = 2$ and both stations are sending a packet, the average number of collision steps is $\overline{C}(2,2) = 1$. We can now compute the average number of collision steps $\overline{C}(3,2)$. Given that $m = 2$, we have to split $n = 3$ into two sub-trees of size $\alpha = 2$ and $\beta = 1$, respectively. Therefore, we can either have two active stations in the α-split and none in the β-split; or one active station in the α-split and one in the β-split. The corresponding probabilities are $P\left\{(2 \in \alpha\text{-split}) \wedge (0 \in \beta\text{-split})\right\} = \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}$ and $P\left\{(1 \in \alpha\text{-split}) \wedge (1 \in \beta\text{-split})\right\} = \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}$. For both cases the probability of the split must be multiply by the average number of collision steps of the right subtree plus the average number of collision steps for the left subtree plus one collision step for the root of both subtrees. Therefore,

$$\begin{aligned}
\overline{C}(3,2) &= P\left\{(2 \in \alpha\text{-split}) \wedge (0 \in \beta\text{-split})\right\} \left[\overline{C}(2,2) + \overline{C}(1,0) + 1\right] + \\
&\quad P\left\{(1 \in \alpha\text{-split}) \wedge (1 \in \beta\text{-split})\right\} \left[\overline{C}(2,1) + \overline{C}(1,1) + 1\right] \\
&= \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}} \left[\overline{C}\,(2,2) + \overline{C}\,(1,0) + 1\right] + \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}} \left[\overline{C}\,(2,1) + \overline{C}\,(1,1) + 1\right] \\
&= \sum_{i=0}^{1} \frac{\binom{2}{2-i}\binom{1}{i}}{\binom{3}{2}} \left[\overline{C}\,(2,2-i) + \overline{C}\,(1,i) + 1\right] \qquad (2.4)
\end{aligned}$$

Now, assume that for all $x \leq n-1$, $\overline{C}(x,m)$ satisfies Eq. (2.3). We need to show that the same holds for $\overline{C}(n,m)$. If n is even then $\alpha = \beta = \frac{n}{2}$, otherwise $\beta = \alpha - 1$. First observe that the probability that $m-i$ active stations are in the α-split while the remaining i active stations are in the β-split is given by

$$P\left\{(m-i \in \alpha\text{-split}) \wedge (i \in \beta\text{-split})\right\} = \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \qquad (2.5)$$

Therefore, the average number of collision steps for the α- and β-split is

$$\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{C}\,(\alpha, m-i) + \overline{C}\,(\beta, i) + 1\right] \qquad (2.6)$$

For the average number of collision steps, $\overline{C}(n,m)$, we need to consider the number of collision steps as well as the probability of each of the possible α- and β-splits. Therefore,

$$\begin{aligned}
\overline{C}(n,m) &= \frac{\binom{\alpha}{m-\mu}\binom{\beta}{\mu}}{\binom{n}{m}} \left[\overline{C}\,(\alpha, m-\mu) + \overline{C}\,(\beta, \mu) + 1\right] + \\
&\quad \frac{\binom{\alpha}{m-\mu-1}\binom{\beta}{\mu+1}}{\binom{n}{m}} \left[\overline{C}\,(\alpha, m-\mu-1)) + \overline{C}\,(\beta, \mu+1) + 1\right] + \ldots\ldots + \\
&\quad \frac{\binom{\alpha}{m-\nu+1}\binom{\beta}{\nu-1}}{\binom{n}{m}} \left[\overline{C}\,(\alpha, m-\nu+1) + \overline{C}\,(\beta, \nu-1) + 1\right] + \\
&\quad \frac{\binom{\alpha}{m-\nu}\binom{\beta}{\nu}}{\binom{n}{m}} \left[\overline{C}\,(\alpha, m-\nu) + \overline{C}\,(\beta, \nu) + 1\right] \qquad (2.7)
\end{aligned}$$

Now, three cases must be considered depending on the relationship between m, α, β. First, when $m \leq \alpha$ and $m \leq \beta$ we have $\mu = 0$ and $\nu = m$. When $m \leq \alpha$ and $\beta < m$ we

have $\mu = 0$ and $\nu = \beta$. Finally, when m is greater than α and β we have $\mu = m - \alpha$ and $\nu = \beta$. Note that, since $\beta \leq \alpha$ the parameter m cannot be at the same time $> \alpha$ and $\leq \beta$. Eq. (2.3) then immediately follows by summing the average number of steps for each of the possible splits. ∎

Theorem 1 gives the average number of collision steps required to resolve all initial m collisions. If we wish to compute the average number of collision steps up to the kth success for all $1 \leq k \leq m \leq n$, then the same approach used to derive Eq. (2.3) yields the following recursion.

$$\overline{C}(n,m,k) = \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \begin{cases} \overline{C}(\alpha, m-i) + \overline{C}(\beta, i, k-m+i) + 1 & \text{if } k > m-i \\ \overline{C}(\alpha, m-i, m-i) + 1 & \text{otherwise} \end{cases} \tag{2.8}$$

Notice that counting the collision steps ends as soon as the kth success step has been achieved. The following theorem computes the average number of collision steps for the special case when the resolution round is ended as soon as the first success is reached, i.e. $k = 1$.

Theorem 2: *Let n be the total number of stations in the network and m the number of stations at the beginning of the collision resolution round. Then for all $1 < m \leq n$, the average number of collision steps incurred until the first success step is given by one of the following cases.*

- *Case 1:* $m \leq \min\{\alpha, \beta\}$.

$$\overline{C}(n,m) = \sum_{i=0}^{m} \left[\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{C}(\alpha, m-i) + 1 \right] + \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}} \overline{C}(\beta, m)$$

- *Case 2:* $\beta < m \leq \alpha$.

$$\overline{C}(n,m) = \sum_{i=0}^{\beta} \left[\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{C}(\alpha, m-i) + 1 \right]$$

- *Case 3:* $m > \max\{\alpha, \beta\}$.

$$\overline{C}(n,m) = \sum_{i=m-\alpha}^{\beta} \left[\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{C}(\alpha, m-i) + 1 \right] \tag{2.9}$$

Proof: It is trivial to show that for all $n \geq 1$, $\overline{C}(n,0) = \overline{C}(n,1) = 0$. Furthermore, when $n = 2$ and both stations are active, then $\overline{C}(2,2) = 1$. Using these identities, we can now compute $\overline{C}(3,2)$ up to the first success. Since $m = 2$, we have to split $n = 3$ into two sub-trees of size $\alpha = 2$ and $\beta = 1$ respectively. Now, we can either have two active stations in the α-split and none in the β-split, or one active station in the α-split and one in the β-split. In the former case we have $P\{(2 \in \alpha\text{-split}) \wedge (0 \in \beta\text{-split})\} = \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}$ while in the latter we have $P\{(1 \in \alpha\text{-split}) \wedge (1 \in \beta\text{-split})\} = \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}$. For both cases, the probability of the split must be multiply by the average number of collision steps of the right subtree plus the average number of collision steps for the left subtree plus one collision step for the root of both subtrees up to the first successful transmission. This yields,

$$\overline{C}(3,2) = \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}\left[\overline{C}\left(2,2\right)+1\right] + \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}\left[\overline{C}\left(2,1\right)+1\right] = \sum_{i=0}^{1}\frac{\binom{2}{2-i}\binom{1}{i}}{\binom{3}{2}}\left[\overline{C}\left(2,2-i\right)+1\right] \quad (2.10)$$

Now, assume that for all $x \leq n-1$, $\overline{C}(x,m)$ is given as in Case 1 or Case 2 or Case 3. We need to show that the same holds for $\overline{C}(n,m)$.

Similarly to the proof of Theorem 1, it is not difficult to see that the probability that $m-i$ active stations are in the α-split and the remaining i active stations are in the β-split is $P\{(m-i \in \alpha\text{-split}) \wedge (i \in \beta\text{-split})\} = \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}$. Therefore, the average number of collision steps for this split is given by

$$\text{pair}\ (m-i \in \alpha \wedge i \in \beta) \quad = \quad \begin{cases} \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}}\left[\overline{C}\left(\alpha,0\right)+\overline{C}\left(\beta,m\right)+1\right] & \text{if } i = m \\ \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\left[\overline{C}\left(\alpha,m-i\right)+1\right] & \text{if } i \neq m \end{cases}$$

To compute the average number of collision steps $\overline{C}(n,m)$ we then need to consider the number of collision steps as well as the probability of each of the possible α- and β-splits. First observe that if the first successful transmission is achieved in the α-split, then the β-split does not need to be considered. There are three possible μ-ν combinations. If $m \leq \min\{\alpha,\beta\}$ then $\mu = 0$ and $\nu = m$. In this case only the split with $i = m$ considers the

β-split. All other split pairs have the first successful transmission within the α-split. On the other hand, if $\beta < m \leq \alpha$ then $\mu = 0$ and $\nu = \beta$. The β-split is eliminated in all split pairs. Finally, if $m > \max\{\alpha, \beta\}$ then $\mu = m - \alpha$ and $\nu = \beta$. Once again only the α-split is consider. The thesis then follows by considering the sum of the average steps. ∎

2.3.3 Average Number of Idle Steps

To compute the average number of idle steps in a network with n stations and m initial collisions, we need to count the number of idle steps for each of the possible permutations of m collisions among the n leaves of the corresponding binary tree.

As for the collision steps, the number of idle steps of a binary tree can be expressed as the number of idle steps of the right subtree, plus the number of collision steps of the left subtree except that now we don't count a plus one for the root. Thus, we have the following equation.

$$\mathcal{Z}_{root\ tree} = \mathcal{Z}_{right\ subtree} + \mathcal{Z}_{left\ subtree} \tag{2.11}$$

For the example described in Fig. 2.1 and its six permutations shown in Table 2.1, the average number of idle steps $\overline{\mathcal{Z}}(4, 2)$ can be expressed by

$$\overline{\mathcal{Z}}(4,2) = \sum_{i=0}^{2} \frac{\binom{2}{2-i}\binom{2}{i}}{\binom{4}{2}} \left[\overline{\mathcal{Z}}\,(2, 2-i) + \overline{\mathcal{Z}}\,(2, i)\right] \tag{2.12}$$

Again, the recursion splits the original tree with $n = 4$ into two sub-trees with $n = 2$, where the ratio $\frac{\binom{2}{2-i}\binom{2}{i}}{\binom{4}{2}}$ is the probability of having $2 - i$ stations with a packet in one subtree and i stations with a packet in the other subtree.

Theorem 3 below extends the above result to the general case $1 < m \leq n$.

Theorem 3: *Let n be the total number of stations and m be the number of stations participating in the tree splitting algorithm. Then for all $1 < m \leq n$, the average number of idle steps required to resolve all m collisions is*

$$\overline{\mathcal{Z}}(n,m) = \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{Z}\,(\alpha, m-i) + \mathcal{Z}\,(\beta, i)\right] \tag{2.13}$$

Proof: It is easy to see that for all $n \geq 1$, $\overline{\mathcal{Z}}(n,0) = 1$ and $\overline{\mathcal{Z}}(n,1) = 0$. Another simple case is when $n = 2$ and both stations wish to request the channel. Since in this case resolving the collisions requires only one collision step and two success steps, the average number of idle steps is $\overline{\mathcal{Z}}(2,2) = 0$. Now, we can compute $\overline{\mathcal{Z}}(3,2)$. Since $m = 2$, we have to split $n = 3$ into two sub-trees of size $\alpha = 2$ and $\beta = 1$, respectively. Thus, we can either have two active stations in the α-split and none in the β-split, or one active station in the α-split and one in the β-split. In the former case $P\left\{(2 \in \alpha\text{-split}) \wedge (0 \in \beta\text{-split})\right\} = \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}$, while in the latter $P\left\{(1 \in \alpha\text{-split}) \wedge (1 \in \beta\text{-split})\right\} = \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}$. We then obtain,

$$\begin{aligned}
\overline{\mathcal{Z}}(3,2) &= P\left\{(2 \in \alpha\text{-split}) \wedge (0 \in \beta\text{-split})\right\}\left[\overline{\mathcal{Z}}(2,2) + \overline{\mathcal{Z}}(1,0) + 1\right] + \\
&\quad P\left\{(1 \in \alpha\text{-split}) \wedge (1 \in \beta\text{-split})\right\}\left[\overline{\mathcal{Z}}(2,1) + \overline{\mathcal{Z}}(1,1) + 1\right] \\
&= \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}\left[\overline{\mathcal{Z}}\,(2,2) + \overline{\mathcal{Z}}\,(1,0)\right] + \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}\left[\overline{\mathcal{Z}}\,(2,1) + \overline{\mathcal{Z}}\,(1,1) + 1\right] \\
&= \sum_{i=0}^{1} \frac{\binom{2}{2-i}\binom{1}{i}}{\binom{3}{2}}\left[\overline{\mathcal{Z}}\,(2,2-i) + \overline{\mathcal{Z}}\,(1,i)\right] \qquad (2.14)
\end{aligned}$$

Now, assume that for all $x \leq n-1$, $\overline{\mathcal{C}}(x,m)$ satisfies Eq. (2.13). We need to show that the same holds for $\overline{\mathcal{C}}(n,m)$.

First observe that the probability that $m-i$ active stations are in the α-split and the remaining i active stations are in the β-split is given by $P\left\{(m-i \in \alpha\text{-split}) \wedge (i \in \beta\text{-split})\right\} = \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}$. Therefore, the average number of idle steps for this split is given by the average number of idle steps for the α-split, plus the average number of idle steps for the β-split, times the probability of such a split, i.e.

$$\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\left[\overline{\mathcal{Z}}\,(\alpha, m-i) + \overline{\mathcal{Z}}\,(\beta, i)\right] \qquad (2.15)$$

In order to compute the average number of idle steps $\overline{\mathcal{Z}}(n,m)$, we need to consider the cost as well as the probability of each of the possible α- and β-splits, thus

$$
\begin{aligned}
\overline{\mathcal{Z}}(n,m) \quad = \quad & \frac{\binom{\alpha}{m-\mu}\binom{\beta}{\mu}}{\binom{n}{m}} \left[\overline{\mathcal{Z}}\,(\alpha, m-\mu) + \overline{\mathcal{Z}}\,(\beta,\mu)\right] + \\
& \frac{\binom{\alpha}{m-\mu-1}\binom{\beta}{\mu+1}}{\binom{n}{m}} \left[\overline{\mathcal{Z}}\,(\alpha, m-\mu-1)) + \overline{\mathcal{Z}}\,(\beta,\mu+1)\right] + \ldots.. + \\
& \frac{\binom{\alpha}{m-\nu+1}\binom{\beta}{\nu-1}}{\binom{n}{m}} \left[\overline{\mathcal{Z}}\,(\alpha, m-\nu+1) + \overline{\mathcal{Z}}\,(\beta,\nu-1)\right] + \\
& \frac{\binom{\alpha}{m-\nu}\binom{\beta}{\nu}}{\binom{n}{m}} \left[\overline{\mathcal{Z}}\,(\alpha, m-\nu) + \overline{\mathcal{Z}}\,(\beta,\nu)\right] \qquad (2.16)
\end{aligned}
$$

The three cases then follows by summing the average number of steps for each of the possible splits where α, β, μ and ν are given by Eq. (2.3). ∎

As done in the previous section, we can modify Eq. (2.13) to compute for all $1 \leq k \leq m \leq n$, the average number of idle steps up to the kth success as follows.

$$
\overline{\mathcal{Z}}(n,m,k) = \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \begin{cases} \overline{\mathcal{Z}}(\alpha, m-i) + \overline{\mathcal{Z}}(\beta, i, k-m+i) & \text{if } k > m-i \\ \overline{\mathcal{Z}}(\alpha, m-i, m-i) & \text{otherwise} \end{cases} \qquad (2.17)
$$

A special case of Eq. (2.17) is when $k = 1$, i.e. when the average number of idle steps are counted only up to the first successful transmission.

Theorem 4: *For all $1 < m \leq n$, the average number of idle steps up to the first-successful transmission is given by one of the following cases:*

- *Case 1: $m \leq \min\{\alpha, \beta\}$.*

$$
\overline{\mathcal{Z}}(n,m) = \sum_{i=0}^{m} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{\mathcal{Z}}\,(\alpha, m-i) + \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}} \overline{\mathcal{Z}}\,(\beta, m)
$$

- *Case 2: $\beta < m \leq \alpha$.*

$$
\overline{\mathcal{Z}}(n,m) = \sum_{i=0}^{\beta} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{\mathcal{Z}}\,(\alpha, m-i)
$$

- *Case 3: $m > \max\{\alpha, \beta\}$.*

$$
\overline{\mathcal{Z}}(n,m) = \sum_{i=m-\alpha}^{\beta} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{\mathcal{Z}}\,(\alpha, m-i) \qquad (2.18)
$$

Proof: For all $n \geq 1$, $\overline{\mathcal{Z}}(n,0) = 1$ and $\overline{\mathcal{Z}}(n,1) = 0$. Since when $n = 2$ and both stations are active, resolving the collisions requires only one collision step and one successful step, $\overline{\mathcal{Z}}(2,2) = 0$. We can now compute $\overline{\mathcal{Z}}(3,2)$. Since $m = 2$, we need to split $n = 3$ into two subtrees of size $\alpha = 2$ and $\beta = 1$ respectively. Therefore, we can either have two active stations in the α-split and none in the β-split, or one active station in the α-split and one in the β-split. In this cases it is not difficult to see that $P\{(2 \in \alpha\text{-split}) \wedge (0 \in \beta\text{-split})\} = \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}$ and $P\{(1 \in \alpha\text{-split}) \wedge (1 \in \beta\text{-split})\} = \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}$. Therefore,

$$\overline{\mathcal{Z}}(3,2) = \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}\overline{\mathcal{Z}}(2,2) + \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}\overline{\mathcal{Z}}(2,1) = \sum_{i=0}^{1} \frac{\binom{2}{2-i}\binom{1}{i}}{\binom{3}{2}}\overline{\mathcal{Z}}(2,2-i) = 0 \tag{2.19}$$

Now, assume that for all $x \leq n-1$, $\overline{\mathcal{Z}}(x,m)$ is given as in Case 1 or Case 2 or Case 3. We need to show that the same holds $\overline{\mathcal{Z}}(n,m)$. Since $P\{(m-i \in \alpha\text{-split}) \wedge (i \in \beta\text{-split})\} = \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}$, the average number of idle steps for this split is given by

$$\text{pair}\ (m-i \in \alpha \wedge i \in \beta) \quad = \quad \begin{cases} \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}}\left[\overline{\mathcal{Z}}\,(\alpha,0) + \overline{\mathcal{Z}}\,(\beta,m)\right] & \text{if } i = m \\ \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\overline{\mathcal{Z}}\,(\alpha,m-i) & \text{if } i \neq m \end{cases}$$

In order to compute the average number of idle steps, $\overline{\mathcal{Z}}(n,m)$, we need to consider the cost as well as the probability of each of the possible α- and β-splits. Theorem 4 then follows by summing the average number of steps for each of the possible splits. ∎

We conclude this section by observing that for all $n \geq 1$ and $m > 1$, $\mathcal{S}(n,m)$, $\mathcal{Z}(n,m)$ and $\mathcal{C}(n,m)$ must satisfy the following identity.

$$\mathcal{S}(n,m) + \mathcal{Z}(n,m) - \mathcal{C}(n,m) - 1 = 0 \tag{2.20}$$

Notice that this equation applies to the average number of steps as well as to any specific ith permutation.

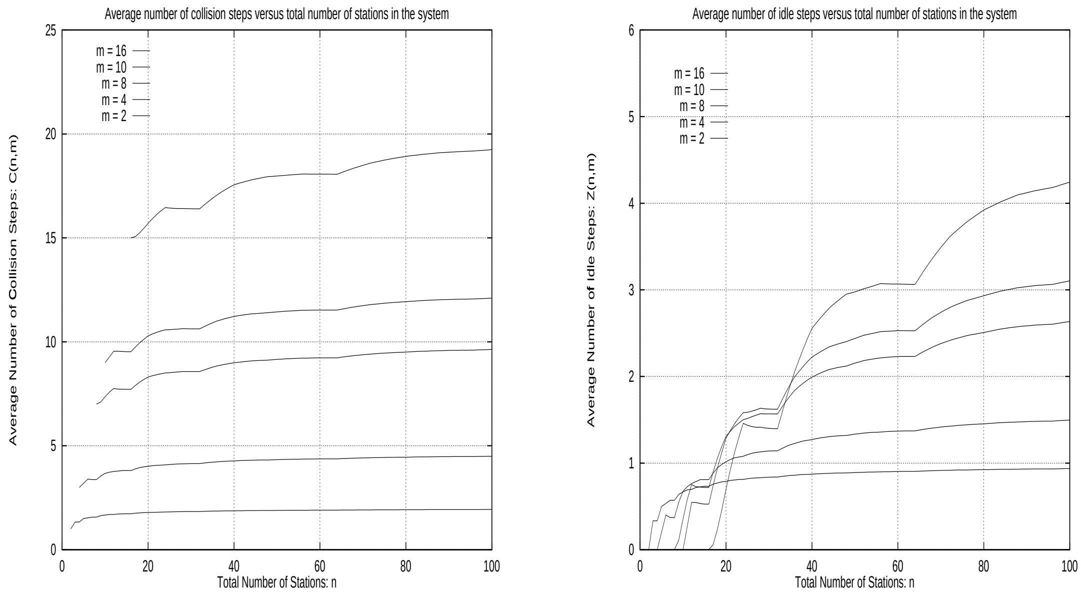

Figure 2.2: Average number of collision and idle steps as a function of the total number of stations in the system.

2.3.4 Average Number of Total Steps

The total average number of steps required to resolve m collision in a system with n stations is the sum of all three types of steps. This sum can also be expressed as the average number of total steps of the right subtree plus the average number of steps of the left subtree plus one for the root. We have thus proved the following theorem.

Theorem 5: *Let n be the total number of stations and m and the number of stations participating in the tree splitting algorithm. Then for all $1 < m \leq n$, the average number of collision-resolution steps required to resolve all m collisions is*

$$\begin{aligned} \mathcal{T}(n,m) &= \mathcal{S}(n,m) + \mathcal{Z}(n,m) + \mathcal{C}(n,m) \\ &= \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{T}(\alpha, m-i) + \mathcal{T}(\beta, i) + 1\right] \end{aligned} \qquad (2.21)$$

Proof Sketch: The proof immediately follows from Theorem 1 and Theorem 3. ∎

Eq. (2.21) can also be extended to include the average number of total steps up to the kth success. That is,

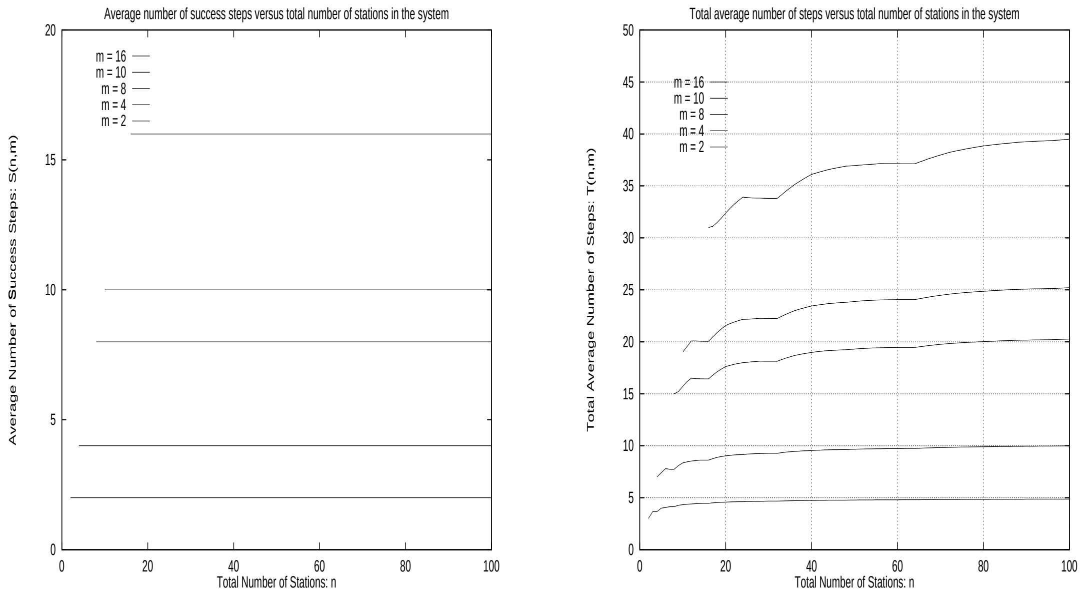

Figure 2.3: Average number of success steps and total number of steps as a function of the total number of stations in the system.

$$\overline{\mathcal{T}}(n,m,k) = \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \begin{cases} \overline{\mathcal{T}}(\alpha, m-i) + \overline{\mathcal{T}}(\beta, i, k-m+i) + 1 & \text{if } k > m-i \\ \overline{\mathcal{T}}(\alpha, m-i, m-i) + 1 & \text{otherwise} \end{cases} \qquad (2.22)$$

In Figs. 2.2 and 2.3 we plotted the average number of collision, idle, success and total steps. Each line represents a fix m. From top to bottom the lines in the graph are for $m = 16, 10, 8, 4, 2$. As shown in Fig. 2.3, for fixed m the total average number of steps, $\mathcal{T}(n,m)$, is strictly monotone in n. Furthermore, the function $\mathcal{T}(n,m)$ increases in n, and

$$\max \mathcal{T}(n,m) = \lim_{n\to\infty} \mathcal{T}(n,m) \qquad (2.23)$$

This equation will be used in the next chapter to prove that the probabilistic tree-splitting algorithm introduced by Capetanakis [9] is an upper bound for our deterministic tree-splitting algorithm.

2.4 Upper Bounds

In this Section we derive upper bounds for the three recursions introduced in the previous Section. Our upper bounds depend *only* on the number m of active stations in the system and are independent of the number n of stations.

With respect to the average number of success steps, it is not difficult to see that $\overline{\mathcal{S}}(n,m) \le m$. In the following theorems we derive upper bounds on the average number of idle and collision steps. Our first guess for the upper bound was derived from the recursive equations by increasing n. Tighter upper bound were then derived by lowering such guess until the proof could no longer hold.

$\mathcal{Z}(1,0) = 1$	$\mathcal{Z}(3,2) = \frac{1}{3} \le 0.886$	$\mathcal{C}(1,0) = 0$	$\mathcal{C}(3,2) = \frac{4}{3} \le 1.886$
$\mathcal{Z}(1,1) = 0$	$\mathcal{Z}(3,3) = 0 \le 1.299$	$\mathcal{C}(1,1) = 0$	$\mathcal{C}(3,3) = 2$
$\mathcal{Z}(2,0) = 1$	$\mathcal{Z}(4,0) = 1$	$\mathcal{C}(2,0) = 0$	$\mathcal{C}(4,0) = 0$
$\mathcal{Z}(2,1) = 0$	$\mathcal{Z}(4,1) = 0$	$\mathcal{C}(2,1) = 0$	$\mathcal{C}(4,1) = 0$
$\mathcal{Z}(2,2) = 0 \le 0.886$	$\mathcal{Z}(4,2) = \frac{1}{3} \le 0.886$	$\mathcal{C}(2,2) = 1 \le 1.886$	$\mathcal{C}(4,2) = \frac{4}{3} \le 1.886$
$\mathcal{Z}(3,0) = 1$	$\mathcal{Z}(4,3) = 0 \le 1.299$	$\mathcal{C}(3,0) = 0$	$\mathcal{C}(4,3) = 2 \le 3.329$
$\mathcal{Z}(3,1) = 0$	$\mathcal{Z}(4,4) = 0 \le 1.732$	$\mathcal{C}(3,1) = 0$	$\mathcal{C}(4,4) = 3 \le 4.772$

Table 2.3: Initial Conditions

Theorem 6: *Let n be the number of stations and m be the number of stations participating in the tree splitting algorithm. Then for all $1 < m \le n$, the average number of collision steps is bounded by*

$$\overline{\mathcal{C}}(n,m) \le 1.443m - 1 \tag{2.24}$$

Proof: The Theorem is proved by induction on the number n of stations. According to Eq. (2.3), $\overline{\mathcal{C}}(n,m)$ can be expressed as

$$\overline{\mathcal{C}}(n,m) = \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m-i\right) + \mathcal{C}\left(\beta, i\right) + 1\right] \tag{2.25}$$

Now, three cases must be considered depending on the relationship between m, α, β.

We start by analyzing the case $m \le \min\{\alpha, \beta\}$. Since $\mu = 0$ and $\nu = m$, Eq. (2.25) can be written as

$$\begin{aligned}
\overline{\mathcal{C}}(n,m) &= \sum_{i=2}^{m-2} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m-i\right) + \mathcal{C}\left(\beta, i\right) + 1\right] + \frac{\binom{\alpha}{m}\binom{\beta}{0}}{\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m\right) + 1\right] + \\
&\quad \frac{\binom{\alpha}{m-1}\binom{\beta}{1}}{\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m-1\right) + 1\right] + \frac{\binom{\alpha}{1}\binom{\beta}{m-1}}{\binom{n}{m}} \left[\mathcal{C}\left(\beta, m-1\right) + 1\right] + \\
&\quad \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}} \left[\mathcal{C}\left(\beta, m\right) + 1\right] \qquad (2.26)
\end{aligned}$$

Since by the inductive hypothesis, for all $\alpha < n$ and $\beta < n$, $\mathcal{C}\left(\beta, i\right) \leq 1.443i - 1$ and $\mathcal{C}\left(\alpha, i\right) \leq 1.443i - 1$, the function $\overline{\mathcal{C}}(n,m)$ can be upper bounded by

$$\overline{\mathcal{C}}(n,m) \leq 1.443m - 1 + \frac{\binom{\alpha}{m}}{\binom{n}{m}} - 1.443\beta\frac{\binom{\alpha}{m-1}}{\binom{n}{m}} - 1.443\alpha\frac{\binom{\beta}{m-1}}{\binom{n}{m}} + \frac{\binom{\beta}{m}}{\binom{n}{m}} \qquad (2.27)$$

Furthermore, by applying the binomial coefficient identity to each of the binomial coefficients in the right hand side of Eq. (2.27), we obtain

$$\begin{aligned}
\overline{\mathcal{C}}(n,m) &\leq 1.443m - 1 + \frac{\alpha - 1.443m\beta + 1 - m}{m} \frac{\binom{\alpha}{m-1}}{\binom{n}{m}} + \\
&\quad \frac{\beta - 1.443m\alpha + 1 - m}{m} \frac{\binom{\beta}{m-1}}{\binom{n}{m}}
\end{aligned} \qquad (2.28)$$

Now, when n is even we have $\alpha = \beta = \frac{n}{2}$ and

$$\overline{\mathcal{C}}(n,m) \leq \frac{3}{2} - 1 + 2\frac{\alpha(1 - 1.443m) + (1 - m)}{m} \frac{\binom{\alpha}{m-1}}{\binom{n}{m}} \leq 1.443m - 1 \qquad (2.29)$$

On the other hand, when n is odd we have $\beta = \alpha - 1$ and

$$\begin{aligned}
\overline{\mathcal{C}}(n,m) \leq 1.443m - 1 &+ \frac{\beta(1 - 1.443m) + (2 - m)}{m} \frac{\binom{\alpha}{m-1}}{\binom{n}{m}} + \\
&\quad \frac{\alpha(1 - 1.443m) - m}{m} \frac{\binom{\beta}{m-1}}{\binom{n}{m}} \leq 1.443m - 1
\end{aligned} \qquad (2.30)$$

Therefore, for all $1 < m \leq n$, we have proved that $\overline{\mathcal{C}}(n,m) \leq 1.443m - 1$.

Next, we consider the case $\beta < m \leq \alpha$. Since $\mu = 0$ and $\nu = \beta$, we have that $\alpha = m$, $\beta = m - 1$ and $n = 2m - 1$. Substituting these values in Eq. (2.3) we obtain,

$$\begin{aligned}\overline{\mathcal{C}}(n,m) &= \sum_{i=2}^{\beta-1} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{m\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m-i\right) + \mathcal{C}\left(\beta, i\right) + 1\right] + \frac{\binom{\alpha}{m}\binom{\beta}{0}}{m\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m\right) + 1\right] \\ &+ \frac{\binom{\alpha}{m-1}\binom{\beta}{1}}{m\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m-1\right) + 1\right] + \frac{\binom{\alpha}{m-\beta}\binom{\beta}{\beta}}{m\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m-\beta\right) + \beta\right] \qquad (2.31)\end{aligned}$$

Noting that by the inductive hypothesis, for all $\alpha < n$ and $\beta < n$, $\mathcal{C}\left(\alpha, i\right) \leq 1.443i - 1$ and $\mathcal{C}\left(\beta, i\right) \leq 1.443i - 1$ we can upper bound the right hand side of Eq. (2.31) simply by

$$\overline{\mathcal{C}}(n,m) \leq 1.443m - 1 + \frac{\binom{\alpha}{m}}{\binom{n}{m}} - 0.443\beta \frac{\binom{\alpha}{m-1}}{\binom{n}{m}} - 1.443\beta \frac{\binom{\alpha}{m-\beta}}{\binom{n}{m}} \qquad (2.32)$$

Furthermore, by applying the binomial identity to the binomial coefficients in the right hand side of Eq. (2.32) we obtain

$$\overline{\mathcal{C}}(n,m) \leq 1.443m - 1 + \frac{\alpha - 0.443\beta + 1 - m}{m} \frac{\binom{\alpha}{m-1}}{\binom{n}{m}} - 1.443\beta \frac{\binom{\alpha}{m-\beta}}{\binom{n}{m}} \qquad (2.33)$$

Since $\beta < m \leq \alpha$ it follows that $\alpha = m$, $\beta = \alpha - 1$ and we obtain

$$\begin{aligned}\overline{\mathcal{C}}(n,m) &\leq 1.443m - 1 + \frac{1.443 - 0.443m}{m} \frac{\binom{\alpha}{m-1}}{\binom{n}{m}} + \\ &\quad (1 - 1.443m) \frac{\binom{\alpha}{m-\beta}}{\binom{n}{m}} \leq 1.443m - 1 \qquad (2.34)\end{aligned}$$

Thus, also in this case it follows that, for any $1 < m \leq n$, $\overline{\mathcal{C}}(n,m) \leq 1.443m - 1$.

Finally we consider $m > \max\{\alpha, \beta\}$. Since in this case we have $\mu = m - \alpha$ and $\nu = \beta$, Eq. (2.3) can be written as

$$\overline{\mathcal{C}}(n,m) = \sum_{i=m-\alpha}^{\beta} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{C}\left(\alpha, m-i\right) + \mathcal{C}\left(\beta, i\right) + 1\right] \qquad (2.35)$$

Notice that terms with $i = 0$ do not appear in the right hand side of Eq. (2.35) and that the smallest i value is 1. Furthermore, since $\mathcal{C}\left(\beta, 1\right) = \mathcal{C}\left(\alpha, 1\right) = 0$, it follows that $\mathcal{C}\left(\beta, 1\right) = \mathcal{C}\left(\alpha, 1\right) \leq 1.443m - 1$ and we obtain,

$$\begin{aligned}
\overline{C}(n,m) &\leq \sum_{i=m-\alpha}^{\beta-1} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}[1.443(m-i)-1+1.443i-1+1]+ \\
&\quad \frac{\binom{\alpha}{m-\beta}\binom{\beta}{\beta}}{\binom{n}{m}}[1.443m-1.443\beta-1+\beta] \\
&= 1.433m-1-0.443\beta\frac{\binom{\alpha}{m-\beta}}{\binom{n}{m}} \leq 1.443m-1 \qquad (2.36)
\end{aligned}$$

This conclude the proof of the Theorem. ∎

Next, we derive an upper bound on the number of collision steps required up to the first success, i.e. $k=1$.

Theorem 7: *Let n be the number of stations and m be the number of initial colliding stations participating in the tree splitting algorithm. Then for all $1 < m \leq n$, the average number of collision steps up to the first success is bounded by*

$$\overline{C}(n,m) \leq \log(m)+1 \qquad (2.37)$$

Proof: The theorem is proved by induction on m. As in the proof of Theorem 2, three cases must be considered.

We start with $m \leq \max\{\alpha,\beta\}$. By applying Theorem 2 and substituting $\overline{C}(\alpha,i)$ with $\log(2i)$ we get

$$\overline{C}(n,m) \leq \log(2) + \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}}\log(2m) + \sum_{i=0}^{m-1}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\log(2(m-i)) \leq \log(2m) \qquad (2.38)$$

On the other hand, when either $\beta < m \leq \alpha$ or $m > \max\{\alpha,\beta\}$ we have

$$\overline{C}(n,m) = 1 + \sum_{i=\mu}^{\nu}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\overline{C}\left(\alpha, m-i\right) \qquad (2.39)$$

Using the fact that $\overline{C}(\alpha,i) \leq \log(2i)$, yields

$$\overline{C}(n,m) \leq \log(2) + \sum_{i=\mu}^{\nu}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\log(2(m-i)) \leq \log(2m) \qquad (2.40)$$

and this conclude the proof of the Theorem. ∎

The next result shows that the average number of idle steps required to resolve m collisions in a system of n stations is also bounded.

Theorem 8: *Let n be the number of stations and m be the number of stations participating in the tree splitting algorithm. Then for all $1 < m \leq n$, the average number of idle steps is bounded by*

$$\overline{\mathcal{Z}}(n,m) \leq 0.443m. \tag{2.41}$$

Proof: The values for $\overline{\mathcal{Z}}(n,m)$ and $\overline{\mathcal{C}}(n,m)$ up to $n = 4$ are given in Table 2.3. Also notice that $\overline{\mathcal{Z}}(n,0) = 1$, $\overline{\mathcal{Z}}(n,1) = 0$ and $\overline{\mathcal{Z}}(n,n) = 0$.

Now assume that, for all $4 \leq n \leq \alpha$ and for all $2 \leq m \leq \nu$, $\overline{\mathcal{Z}}(\alpha,m) \leq 0.443m$ holds. We need to show that the condition also holds for $\overline{\mathcal{Z}}(n,m)$. Also here three cases must be considered depending on the relationship between m, α and β.

Case 1: $m \leq \min\{\alpha,\beta\}$. Since $\mu = 0$ and $\nu = m$, it follows that

$$\frac{\overline{\mathcal{Z}}(n,m)}{m} = \sum_{i=0}^{m} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{Z}\left(\alpha, m-i\right) + \mathcal{Z}\left(\beta, i\right)\right] \tag{2.42}$$

Extracting the first and the last two terms in the summation (i.e., the elements with $i = 0, 1, m-1, m$) and noting that $\overline{\mathcal{Z}}(\alpha,1) = \overline{\mathcal{Z}}(\beta,1) = 0$, and $\overline{\mathcal{Z}}(\alpha,0) = \overline{\mathcal{Z}}(\beta,0) = 1$, we obtain

$$\begin{aligned}\overline{\mathcal{Z}}(n,m) \quad = \quad & \sum_{i=2}^{m-2} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{Z}\left(\alpha, m-i\right) + \mathcal{Z}\left(\beta, i\right)\right] + \frac{\binom{\alpha}{m}\binom{\beta}{0}}{\binom{n}{m}} \left[\mathcal{Z}\left(\alpha, m\right) + 1\right] + \\ & \frac{\binom{\alpha}{m-1}\binom{\beta}{1}}{\binom{n}{m}} \mathcal{Z}\left(\alpha, m-1\right) + \frac{\binom{\alpha}{1}\binom{\beta}{m-1}}{\binom{n}{m}} \mathcal{Z}\left(\beta, m-1\right) + \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}} \left[1 + \mathcal{Z}\left(\beta, m\right)\right]\end{aligned} \tag{2.43}$$

Since $\overline{\mathcal{Z}}(\alpha,m) \leq 0.443m$ and $\overline{\mathcal{Z}}(\beta,m) \leq 0.443m$, we further obtain

$$\begin{aligned}\overline{\mathcal{Z}}(n,m) &\leq 0.443m\sum_{i=0}^{m}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}+\frac{\binom{\alpha}{m}\binom{\beta}{0}}{\binom{n}{m}}- \\ &\quad 0.443\frac{\binom{\alpha}{m-1}\binom{\beta}{1}}{\binom{n}{m}}-0.443\frac{\binom{\alpha}{1}\binom{\beta}{m-1}}{\binom{n}{m}}+\frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}} \qquad (2.44)\end{aligned}$$

By applying the identity $\binom{n}{k}=\frac{n-k+1}{k}\binom{n}{k-1}$ to the binomial coefficients occuring in the right hand side of Eq. (2.44) and noticing that the sum equals one we have that

$$\begin{aligned}\overline{\mathcal{Z}}(n,m) &\leq 0.443m+\frac{\alpha-0.443m\beta+1-m}{m}\frac{\binom{\alpha}{m-1}}{\binom{n}{m}}+ \\ &\quad \frac{\beta-0.443m\alpha+1-m}{m}\frac{\binom{\beta}{m-1}}{\binom{n}{m}} \qquad (2.45)\end{aligned}$$

Now, if we can show that the last two terms in Eq. (2.45) are ≤ 0, then it follows that $\overline{\mathcal{Z}}(n,m)\leq 0.443m$. When n is even we have $\alpha=\beta$ and

$$\begin{aligned}\overline{\mathcal{Z}}(n,m) &\leq 0.443m+\frac{(2-0.886m)\alpha+(2-2m)}{m}\frac{\binom{\alpha}{m-1}}{\binom{n}{m}} \\ &\leq 0.443m\end{aligned}$$

where the last inequality follows from the fact that for $m>3$, $\alpha(2-0.886m)+(2-2m)<0$. When n is odd, we have $\beta=\alpha-1$ and again

$$\begin{aligned}\overline{\mathcal{Z}}(n,m) &\leq 0.443m+\frac{\alpha(1-0.443m)+1-0.557m}{m}\frac{\binom{\alpha}{m-1}}{\binom{n}{m}}+ \\ &\quad \frac{\alpha(1-0.443m)-m}{m}\frac{\binom{\beta}{m-1}}{\binom{n}{m}} \\ &\leq 0.443m\end{aligned}$$

where the last inequality follows from the fact that for $m>3$, $\alpha(1-0.443m)+1-0.557m\leq 0$ and $\alpha(1-0.443m)-m\leq 0$.

Case 2: $\beta<m\leq\alpha$. Since $\mu=0$, $\nu=\beta$, $\alpha=m$, $\beta=m-1$ and $n=2m-1$, by extracting the first term in the summation in Eq. (2.13) we obtain

$$\overline{\mathcal{Z}}(n,m) = \sum_{i=1}^{\beta} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\mathcal{Z}(\alpha, m-i) + \mathcal{Z}(\beta, i)\right] + \frac{\binom{\alpha}{m}\binom{\beta}{0}}{\binom{n}{m}} \tag{2.46}$$

Now, since $\mathcal{Z}(m, m-i) \leq 0.443(m-i)$ and $\mathcal{Z}(m-1, i) \leq 0.443i$, it follows that

$$\overline{\mathcal{Z}}(n,m) \leq \sum_{i=1}^{\beta} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} [0.443(m-i) + 0.443i] + \frac{\binom{\alpha}{m}\binom{\beta}{0}}{\binom{n}{m}} = 0.443m \tag{2.47}$$

Case 3: $m > \max\{\alpha, \beta\}$. Since in this case $\mu = m - \alpha$ and $\nu = \beta$, once again, by extracting the last term in the summation in Eq. (2.13) and observing that $\mathcal{Z}(\alpha, m-i) \leq 0.443(m-i)$ and $\mathcal{Z}(\beta, i) \leq 0.443i$, we obtain

$$\overline{\mathcal{Z}}(n,m) = 0.443m - 0.443\beta \frac{\binom{\alpha}{m-\beta}\binom{\beta}{m}}{\binom{n}{m}} \leq 0.443m \tag{2.48}$$

and this conclude the proof of the Theorem. ∎

Finally, in the next result we prove that also the average number of idle steps up to the first success is bounded.

Theorem 9: *Let n be the number of stations and m be the number of initial colliding stations participating in the tree splitting algorithm. Then for all $1 < m \leq n$, the average number of idle steps up to the first success is bounded by*

$$\overline{\mathcal{Z}}(n,m) \leq \frac{1}{2} \tag{2.49}$$

Proof: From Table 2.2, we know that $\overline{\mathcal{Z}}(n,0) = 1$, $\overline{\mathcal{Z}}(n,1) = 0$ and $\overline{\mathcal{Z}}(n,n) = 0$. Furthermore, when $n = 2$ and $m = 2$, we have $\overline{\mathcal{Z}}(2,2) = 0 \leq \frac{1}{2}$.

Now assume that for all $2 \leq n \leq \alpha$ and for all $2 \leq m \leq \nu$, $\overline{\mathcal{Z}}(\alpha, m) \leq \frac{1}{2}$ holds. We need to show that the condition also holds for $\overline{\mathcal{Z}}(n,m)$. There are three cases to be considered.

Case 1: $m \leq \min\{\alpha, \beta\}$. Since $\mu = 0$ and $\nu = m$, using Theorem 4 and substituting every $\overline{\mathcal{Z}}$ with $\frac{1}{2}$ we get

$$\overline{\mathcal{Z}}(n,m) \leq \frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}} + \frac{1}{2}\frac{\binom{\alpha}{0}\binom{\beta}{m}}{\binom{n}{m}} + \frac{1}{2}\sum_{i=0}^{m-2} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \tag{2.50}$$

Using the identity $\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$ we obtain

$$\overline{\mathcal{Z}}(n,m) \leq \frac{1}{2} + \frac{2\beta - m\alpha + 2(1-m)}{2m} \frac{\binom{\beta}{m-1}}{\binom{n}{m}} \leq \frac{1}{2} \tag{2.51}$$

The proofs of Case 2 ($m \leq \alpha$ and $m > \beta$) and Case 3 ($m > \alpha$ and $m > \beta$) can be done together starting with

$$\begin{aligned} \overline{\mathcal{Z}}(n,m) &= \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{\mathcal{Z}}\,(\alpha, m-i) \\ &\leq \sum_{i=\mu}^{\nu} \frac{1}{2} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \leq \frac{1}{2} \end{aligned}$$

where the inequality follows by applying the inductive hypothesis to $\overline{\mathcal{Z}}(\alpha, m)$. Thus, we have proved that for all $m > 2$ and $n \geq 1$, $\overline{\mathcal{Z}}(n,m) \leq \frac{1}{2}$. ∎

The upper bounds derived in Theorems 6 and 8 gives us the worst upper case on the average number of steps that the probabilistic tree-splitting algorithm requires to resolve collisions. An interesting aspect of our bounds is that they are independent of the number n of stations in the network and only depends on the number m of collisions. In particular, we have that the total number of collision-resolution steps needed to resolve m collisions is smaller than $2.886m - 1$ steps.

2.5 Summary

In this chapter we have presented the deterministic tree-splitting algorithm. For each of the three types of collision/idle/success steps we have derived a recursive equation for computing the number of steps required to resolve all collisions. We have also derived recursive equations for the number of steps required up to the kth success step. Furthermore, we have proved upper bounds for each of the recursive equations presented in the chapter. These results are extremely important since the deterministic tree-splitting algorithm is used by all the protocols to resolve collisions.

3. Performance Comparisons

In this chapter the performance of the deterministic tree-splitting algorithm introduced in Chapter 2 is compared with the performance of three well known algorithms: slotted ALOHA [44], the probabilistic tree splitting algorithm proposed independently by Capetanakis [9], Tsybakov and Mikhailovic [54] and Hayes [27], and DQRAP [57] which is a competitive protocol based on collision resolution. The comparison with slotted ALOHA shows that when the number m of collisions is greater than 4, then our deterministic tree-splitting algorithm outperforms slotted ALOHA, although the latter exhibits a slightly better performance when $m < 4$. With respect to the probabilistic tree-splitting algorithm, we show that the average number of steps required by the probabilistic algorithm is larger than the average number of steps required by the deterministic one. More precisely, we show that when we let the number n of stations approach infinity, then the deterministic model converges from below to the probabilistic model. Finally, we show that the deterministic tree-splitting algorithm also outperforms the DQRAP algorithm.

The chapter is organized as follows. In the next section we briefly review the probabilistic tree-splitting algorithm and provide a closed form for the average number of steps required by the algorithm to resolve m collisions. The comparison between the probabilistic and deterministic tree-splitting algorithms is then described in Section 3.2. In Section 3.3 we analyze the approximate throughput of the deterministic tree-splitting algorithm when a slotted channel is assumed and compare it with the throughput of slotted ALOHA. Finally, in Section 3.4 the deterministic tree-splitting algorithm is compared with DQRAP.

3.1 Probabilistic Tree-splitting Algorithm

The probabilistic tree-splitting algorithm proposed independently by Capetanakis [9], Tsybakov and Mikhailovic [54], and Hayes [27] is perhaps the most basic collision resolution algorithm. The algorithm is slot based and uses data packets to resolve collisions. If in the first slot m stations collide, then all the stations that are not involved in the collision have

to wait until all the collisions are resolved. The m stations involved in the collision each flips a coin and split randomly into two sets: the *head* set which contains all the stations that flipped head, and the *tail* set which contains all the stations that flipped tail. The station in the tail set have to wait until all the collisions among the stations in the head set are resolved. All the nodes in the head set retransmit in the next slot. In case of a collision, the process is repeated until all collisions are resolved.

The probabilistic tree-splitting algorithm assigns a stack to each node, and each node monitors the channel in order to know when it is allowed to retransmit. New packets arriving during the collision resolution interval have to be backed up until the collision resolution interval is resolved.

We now present an analysis of the probabilistic tree-splitting algorithm which is based on the analysis originally presented by Massey [40].

Let $\overline{L}(m)$ be the average length in slots of a collision resolution interval given that in its first slot there was a collision among m packets. The algorithm requires exactly one step either when one packet or no packet is transmitted in the first slot. Therefore,

$$\overline{L}(0) = \overline{L}(1) = 1 \tag{3.1}$$

Now, if there is a collision in the first slot, i.e. $m \geq 2$, then the algorithm divides the m packets into two sets with probability p and $(1-p)$ of ending in each one of the two sets. Let $\mathcal{Q}_i(m)$ be the probability that exactly i out of m colliding packets flip heads, and thus need to retransmit in the next slot. Clearly, we have that

$$\mathcal{Q}_i(m) = \binom{m}{i} p^i (1-p)^{m-i} \qquad 0 \leq \mathrm{i} \leq \mathrm{m} \tag{3.2}$$

It is not difficult to see that the average number of steps required to resolve m colliding packets is,

$$\overline{L}(m) \quad = \quad 1 + \sum_{i=0}^{m} \mathcal{Q}_i(m) \left[\overline{L}(i) + \overline{L}(m-i)\right]$$

$$= \quad 1 + \sum_{i=0}^{m} \binom{m}{i} p^i (1-p)^{m-i} \left[\overline{L}(i) + \overline{L}(m-i)\right] \qquad m \geq 2 \tag{3.3}$$

where the first term in Eq. (3.3) corresponds to the slot of the initial collision among m nodes, while the sum corresponds to each of the possible splits times the probability of such a split. Observe that it takes on the average $\overline{L}(i)$ steps to resolve the ith split and $\overline{L}(m-i)$ steps to resolve the $(m-i)$th split. Since $\overline{L}(m)$ appears on both sides of the Eq. (3.3), by solving for $\overline{L}(m)$ we obtain the recursive equation

$$\overline{L}(m) = \frac{1 + \sum_{i=0}^{m-1} \left[\mathcal{Q}_i(m) + \mathcal{Q}_{m-i}(m)\right] \overline{L}(i)}{1 - \mathcal{Q}_0(m) + \mathcal{Q}_m(m)} \qquad m \geq 2 \tag{3.4}$$

It is possible to obtain a close-form expression for this recursion. The derivation is very lengthy[47], and the resulting expression is:

$$\overline{L}(m) = 1 + \sum_{i=2}^{m} \binom{m}{i} \frac{2(i-1)(-1)^i}{1 - p^i - (1-p)^i} \qquad m \geq 2 \tag{3.5}$$

By differentiating Eq. (3.5) with respect to p and setting its derivative to zero, i.e.

$$\frac{\partial \overline{L}(m)}{\partial p} = \sum_{i=2}^{m} \binom{m}{i} \left(2(i-1)(-1)^i\right) \frac{i(p^{i-1} - (1-p)^{i-1})}{(-1 + p^i + (1-p)^i)^2} = 0 \qquad m \geq 2 \tag{3.6}$$

we obtain that the only real value satisfying Eq. (3.6) is $p = \frac{1}{2}$. Since the second partial derivative of Eq. (3.5) is positive for $p = \frac{1}{2}$, it follows that $\overline{L}(m)$ is minimized for $p = \frac{1}{2}$. Thus, for the probabilistic tree-splitting algorithm, on the average, the least number of steps needed to resolve m collisions can be obtain with a fair coin, i.e. $p = \frac{1}{2}$. By setting $p = 1/2$ in Eq. (3.5) we obtain,

$$\overline{L}(m) = 1 + \sum_{i=2}^{m} \binom{m}{i} \frac{2^{i+1}(i-1)(-1)^i}{2^i - 2} \qquad m \geq 2 \tag{3.7}$$

3.2 Probabilistic versus Deterministic Tree Splitting

Having derived the recursive expressions for the probabilistic and for the deterministic tree-splitting protocols (see Eqs. (3.7) and (2.21) respectively), we are now ready to show that the deterministic protocol converges from below to the probabilistic one. Fig. 3.1

compares the total average number of steps required to resolve m initial collisions, with various values of n for the deterministic and probabilistic tree-splitting algorithms. As can be seen in Fig. 3.1, the total average number of steps $\overline{\mathcal{T}}(n,m)$ is always smaller than the total average number of steps required by the probabilistic model. It is clear from Fig. 3.1 that the total average number of steps $\overline{\mathcal{T}}(n,m)$ for the deterministic model is a monotonically increasing function of n.

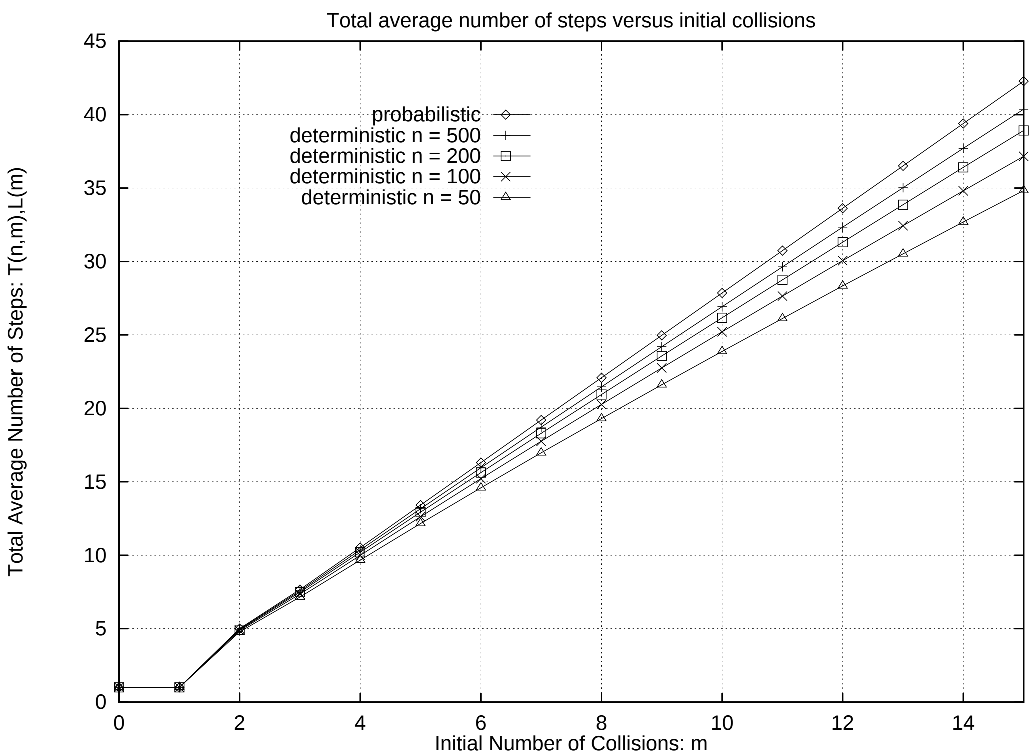

Figure 3.1: The total average number of steps required to resolve m initial collisions for the deterministic tree-splitting algorithm with different n values versus the probabilistic tree-splitting algorithm.

The following lemma investigates the relationship between the distribution probabilities of the deterministic and of the probabilistic tree splitting models, i.e. the probability that i out of m are in the α-split and $m-i$ are in the β-split. This lemma is important because it shows that the probability distribution for the deterministic model converges to that of the probabilistic model when n goes to infinity.

Lemma 1: *For all $0 \leq m \leq n$ and $i \leq \beta$, the coefficients of the probability distribution of the deterministic model is equal to the coefficients of the probabilistic model when n goes to infinity and m is fixed. That is,*

$$\lim_{n\to\infty} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} = \frac{\binom{m}{i}}{2^m} = \mathcal{Q}_i(m) \tag{3.8}$$

Proof: The lemma is proved by induction on the number m of initial collisions. The thesis holds when $m = 0$ and $m = 1$. In fact, when $m = 0$ we have $i = 0$ and thus,

$$\lim_{n\to\infty} \frac{\binom{\alpha}{0}\binom{\beta}{0}}{\binom{n}{0}} = 1\frac{\binom{0}{0}}{2^0} = \mathcal{Q}_{i=0}(m=0) = 1 \tag{3.9}$$

On the other hand, when $m = 1$ we have either $i = 0$ or $i = 1$. By setting $i = 0$, we get

$$\lim_{n\to\infty} \frac{\binom{\alpha}{1}\binom{\beta}{0}}{\binom{n}{1}} = \lim_{n\to\infty} \frac{\alpha}{n} = \frac{1}{2} = \mathcal{Q}_{i=0}(m=1) = \frac{1}{2} \tag{3.10}$$

Now, assume that the lemma holds for $m-1$ where $m \geq 2$. We need to show that the lemma also holds for m. By using the inductive hypothesis, Eq. (3.8) can equivalently be written as

$$\lim_{n\to\infty} \frac{\binom{\alpha}{m-1-i}\binom{\beta}{i}}{\binom{n}{m-1}} = \frac{\binom{m-1}{i}}{2^{m-1}} \tag{3.11}$$

Now, using the fact that $\binom{n}{k} = \frac{k+1}{n-k}\binom{n}{k+1}$, we obtain by simple algebraic manipulation

$$\begin{aligned}
\lim_{n\to\infty} \frac{\binom{\alpha}{m-1-i}\binom{\beta}{i}}{\binom{n}{m-1}} &= \lim_{n\to\infty} \frac{(m-i) + \frac{(m-i)(1-m)}{n}}{\frac{m\alpha}{n} + \frac{m(1+i-m)}{n}} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \\
&= \lim_{n\to\infty} \frac{(m-i) + \frac{(m-i)(1-m)}{n}}{\frac{m\alpha}{n} + \frac{m(1+i-m)}{n}} \lim_{n\to\infty} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \\
&= \frac{2(m-i)}{m} \lim_{n\to\infty} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}
\end{aligned} \tag{3.12}$$

Rearranging Eq. (3.12) we then obtain that

$$\lim_{n\to\infty} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} = \frac{\frac{m}{m-i}}{2} \frac{\binom{m-1}{i}}{2^{m-1}} = \frac{\binom{m}{i}}{2^m} \tag{3.13}$$

and this concludes the proof. ∎

Next, we prove that the average number of steps of the probabilistic model is an upper bound on the average number of steps of the deterministic model.

Theorem 10: *For all $0 \leq m \leq n$, the average number of collision-resolution steps in the deterministic model is bounded above by the average number of collision-resolution steps of the probabilistic model, that is*

$$\lim_{n\to\infty} \mathcal{T}(n,m) = L(m) \tag{3.14}$$

Proof: The proof is by induction on the number of initial collisions m. We first show that the statement is true for $m = 0$ and $m = 1$. This easily follows from the fact that $\mathcal{T}(n,0) = \mathcal{T}(n,1) = 1$. Now, assume that Eq. (3.14) holds for $m-1$ where $m \geq 2$. We need to show that it also holds for m. Using Theorem 5 we can rewrite Eq. (2.21) as

$$\mathcal{T}(n,m) \quad = \quad 1 + \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \mathcal{T}(\alpha, m-i) + \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \mathcal{T}(\beta, i) \tag{3.15}$$

where $\alpha = \lceil n/2 \rceil$, $\beta = n - \alpha$, and the parameters μ and ν depends on the relationship between m, α and β. Thus, three cases need to be considered depending on the value of m, α, β.

- Case 1: $m \leq \min\{\alpha, \beta\}$. The summation goes from $\mu = 0$ to $\nu = m$.
- Case 2: $\alpha \leq< \beta$. The summation goes from $\mu = 0$ to $\nu = \beta$.
- Case 3: $m > \max\{\alpha, \beta\}$. The summation goes from $\mu = m - \alpha$ to $\nu = \beta$.

Since when n goes to infinity we have $\alpha \geq \beta \gg m$, only Case 1 need to be considered. In this case, by using Eq. (3.15) we obtain by simple manipulation,

$$\begin{aligned} \overline{\mathcal{T}}(n,m) \quad - \quad & \frac{\binom{\alpha}{m}}{\binom{n}{m}} \overline{\mathcal{T}}(\alpha, m) - \frac{\binom{\beta}{m}}{\binom{n}{m}} \overline{\mathcal{T}}(\beta, m) = \\ & 1 + \sum_{i=1}^{m} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{\mathcal{T}}(\alpha, m-i) + \sum_{i=0}^{m-1} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \overline{\mathcal{T}}(\beta, i) \end{aligned} \tag{3.16}$$

By applying the limit for n that goes to infinity to both sides we get

$$\lim_{n\to\infty}\left[\overline{\mathcal{T}}(n,m) - \frac{\binom{\alpha}{m}}{\binom{n}{m}}\overline{\mathcal{T}}(\alpha,m) - \frac{\binom{\beta}{m}}{\binom{n}{m}}\overline{\mathcal{T}}(\beta,m)\right] =$$
$$\lim_{n\to\infty}\left[1 + \sum_{i=1}^{m}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\overline{\mathcal{T}}(\alpha,m-i) + \sum_{i=0}^{m-1}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\overline{\mathcal{T}}(\beta,i)\right] \tag{3.17}$$

Now, since $\overline{\mathcal{T}}$ is a monotonically increasing function of n, the terms can be separated as follows

$$\lim_{n\to\infty}\overline{\mathcal{T}}(n,m) - \lim_{n\to\infty}\frac{\binom{\alpha}{m}}{\binom{n}{m}}\lim_{n\to\infty}\overline{\mathcal{T}}(\alpha,m) - \lim_{n\to\infty}\frac{\binom{\beta}{m}}{\binom{n}{m}}\lim_{n\to\infty}\overline{\mathcal{T}}(\beta,m) =$$
$$\lim_{n\to\infty} 1 + \sum_{i=1}^{m}\lim_{n\to\infty}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\lim_{n\to\infty}\overline{\mathcal{T}}(\alpha,m-i) + \sum_{i=0}^{m-1}\lim_{n\to\infty}\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}\lim_{n\to\infty}\overline{\mathcal{T}}(\beta,i) \tag{3.18}$$

Finally, using Lemma 1 and the inductive hypothesis we have that

$$[1 - Q_0(m) - Q_m(m)]\lim_{n\to\infty}\overline{\mathcal{T}}(n,m) = 1 + \sum_{i=1}^{m} Q_i(m)\overline{\mathcal{L}}(m-i) + \sum_{i=0}^{m-1} Q_i(m)\overline{\mathcal{L}}(i) \tag{3.19}$$

or equivalently,

$$\lim_{n\to\infty}\overline{\mathcal{T}}(n,m) = \frac{1 + \sum_{i=1}^{m} Q_i(m)\overline{\mathcal{L}}(m-i) + \sum_{i=0}^{m-1} Q_i(m)\overline{\mathcal{L}}(i)}{[1 - Q_0(m) - Q_m(m)]} = \overline{\mathcal{L}}(m) \tag{3.20}$$

and this concludes the proof. ∎

We have proved that the average number of steps required by the probabilistic model can never be smaller than the average number of steps required by the deterministic model. This is a very important result, because both models have the same complexity.

3.3 Approximate Throughput

Given that the upper bounds on the average number of success/collision/idle steps in a collision-resolution period are independent of the number n of stations, the traffic in the channel can be approximated by assuming

As1) an infinite number of stations, each having at most one packet to send at any time,

As2) a Poisson source for the packets with an aggregate mean generation rate of λ packets per unit time.

Under these assumptions, the average number of packet arrivals for an interval of length $T = \delta$ is clearly $m = \lambda\delta$. Observe that all data blocks have a duration of δ seconds. Throughout the section we will further assume that

As3) data packets can only be sent at the beginning of a slot and that

As4) the size of the slot is equal to the size of a data packet.

Under assumptions **As1** to **As4**, the average channel throughput is easily seen to be

$$S = \frac{\overline{U}}{\overline{B} + \overline{I}} \tag{3.21}$$

where $\overline{U}$ is the average utilization time of the channel, during which the channel is used to transmit data packets, $\overline{B}$ is the expected duration of a busy period, during which the channel is busy with successful or tree-transmission periods, and $\overline{I}$ is the average idle period, i.e., the average interval between two consecutive busy periods. We are now ready to present the main result of the section.

Theorem 11: *Let S be the throughput of the deterministic tree-splitting algorithm and m be the number of initial collisions. Then, we have that*

$$S \geq \frac{m(me^{-m} - 1)}{2.866m(me^{-m} - 1) + 0.866e^{-m}(m - 1)} \tag{3.22}$$

Proof: A packet is said to be successful if during its transmission time it is the only packet in the slot. Thus, the probability of a successful packet is equal to the probability that no arrivals occur in δ seconds. Given that the arrivals of packets to the channel are Poisson distributed with parameter λ, the probability of success is

$$P_s = P\{k = 1 \text{ arrival in a slot}|\text{some arrivals in a slot}\} = \frac{m \cdot e^{-m}}{1 - e^{-m}} \tag{3.23}$$

Under assumption **As1** to **As4**, the average number of stations that participate in the collision-resolution phase is $m = \lambda\delta$. As seen in the previous chapter, each one of the three cases of a collision resolution has an average upper bound on the number of steps that depends on the number m of stations trying to transmit but that it is independent of the total number n of stations. Furthermore, since a busy period is composed of a successful period and tree transmission periods, the duration of an average busy period is given by the sum of the percentage of successful transmission periods per the duration of a transmission period, plus the percentage of the tree periods per their duration. Using the upper bounds derived in Section 2.4, it is not difficult to see that the average busy period can be upper bounded as follows

$$\overline{B} \leq P_s + (m + (1.433 * m - 1) + 0.433 * m)\,(1 - P_s) = P_s + (2.866m - 1)(1 - P_s) \quad (3.24)$$

Observe that, the channel carries user data for δ seconds each time a successful transmission without collision resolution is achieved, and δ seconds for each successful transmission during a collision resolution period. By normalizing the size of a data packet to $\delta = 1$, we obtain that the average utilization is

$$\overline{U} = \delta \cdot P_s + \delta m(1 - P_s) \quad (3.25)$$

Since the average idle period is equal to the average inter-arrival time, we have that

$$\overline{I} \leq \frac{1}{1 - e^{-m}} \quad (3.26)$$

The thesis then immediately follows by substituting Eqs. (3.24), (3.25) and (3.26) into Eq. (3.21). ∎

In Fig. 3.2 we have plotted the throughput per offer load m for the deterministic tree-splitting algorithm and for slotted ALOHA. The throughput $S = m \cdot e^{-m}$ for slotted ALOHA was first derived by Roberts [44]. As we can see in Fig. 3.2, the deterministic tree-splitting algorithm behaves very similar to slotted ALOHA up to $m = 1$. In this region the lower

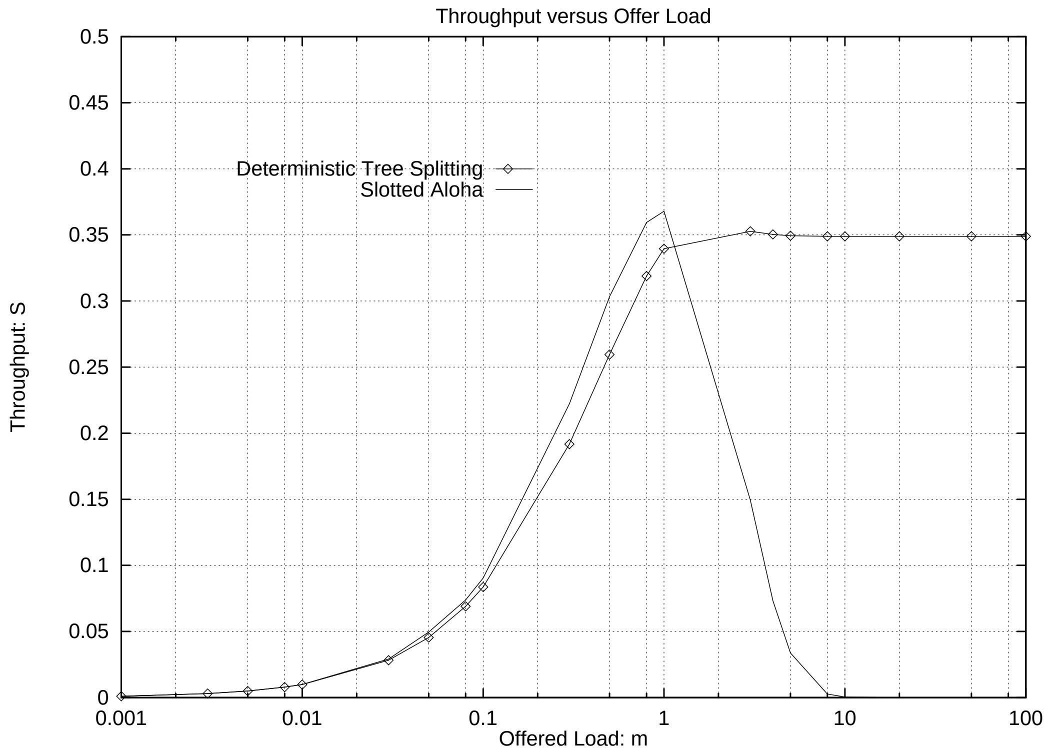

Figure 3.2: Throughput versus offer load m for the deterministic tree-splitting algorithm.

throughput can be explained by observing that, since every success requires on average three steps, resolving fewer collisions is a little worse. In this region a backup mechanism is more efficient than resolving the collisions with the deterministic tree. However, the throughput of the deterministic tree-splitting algorithm increases as the offer load increases up to $m = 4$ and than it slightly decreases maintaining a constant value. It is in this range that the deterministic algorithm out performs slotted ALOHA. In general the result is not very encouraging, since we can never reach a throughput better than $\frac{1}{e}$. A similar result was obtain by Capetanakis [9]. It is clear from the derivations of this chapter that the deterministic tree-splitting algorithm alone does not provide an acceptable throughput. This motivates the use of small control packets to resolve collisions before data packets are sent.

3.4 DQRAP versus Deterministic Tree Splitting

In this section we assume that each step has a cost of one slot, regardless of its type. The average number of steps needed in DQRAP was calculated using the result for collision resolution interval (CRI) length $L(m,s)$ [57] as shown in Eq. (3.27). $L(m,s)$ is a function of the number of station colliding in the previous CRI (m) and the number s of mini-slots per CRI.

$$L(m,s) = \frac{s^{m-1}}{s^{m-1}-1}\left[1+\sum_{k=2}^{m-1}\binom{m}{k}\frac{(s-1)^{m-k}}{s^m}L(k,s)\right] \tag{3.27}$$

Since the number of mini-slots per CRI is s, the total number of steps needed in DQRAP to solve m collisions is equal to the number of mini-slots per CRI times $L(m,s)$.

Fig. 3.3 shows the total number of steps needed to resolve m collisions for both DQRAP and the deterministic tree-splitting algorithm. We have assumed a system with $n = 20$ stations. DQRAP was calculated using different number of mini-slots per CRI.

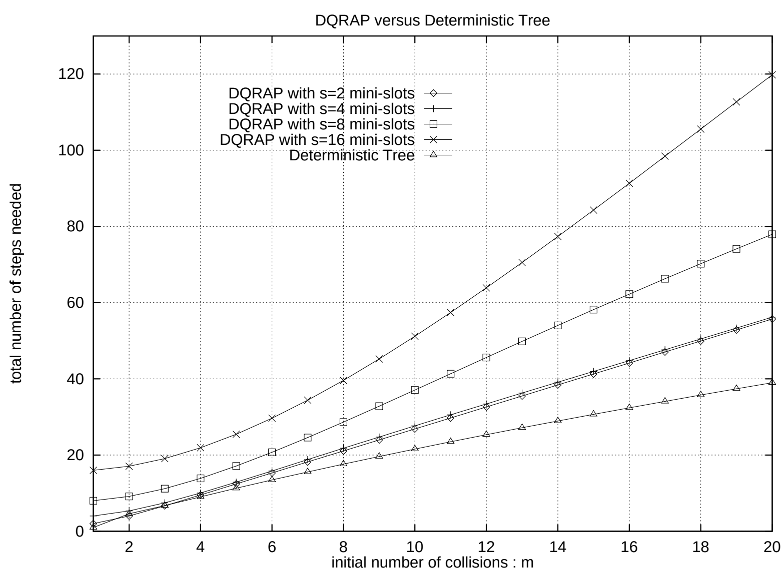

Figure 3.3: Total average number of collision resolution steps for DQRAP with different mini-slots, s=2,4,8,16 compared to the probabilistic tree-splitting algorithm.

As can be seen in Fig. 3.3, the deterministic tree-splitting algorithm outperforms DQRAP.

3.5 Summary

In this chapter we have compared the performance of the deterministic tree-splitting algorithm with the performance of slotted ALOHA, DQRAP and of the probabilistic tree-splitting algorithm. We have shown that the deterministic tree-splitting algorithm outperforms each of these three algorithms. The main contribution of the chapter was to show that the deterministic model converges from below to the probabilistic model, when the number n of stations approaches infinity.

4. CARMA

In the previous chapter, we introduced the deterministic tree-splitting algorithm and compared its performance against slotted ALOHA, the probabilistic tree-splitting algorithm and DQRAP. Even though the deterministic tree-splitting algorithm outperformed each of these algorithms, the throughput results left much to be desired. In this chapter we show that by using control packets instead of the data packets to resolve collisions in a deterministic tree-splitting based protocol the throughput is substantially improved.

Several medium access control (MAC) protocols have been proposed over the past few years [4, 30, 36, 50]. Most of these protocols are based on three- or four-way handshake procedures meant to reduce the number of collisions among data packets, thereby providing better performance than the basic ALOHA or CSMA protocols. The concept of "floor acquisition" was introduced by Fullmer and Garcia-Luna-Aceves [16, 13] for MAC protocols based on such handshake procedures. In a single-channel network, floor acquisition multiple access (FAMA) entails allowing only one station at a time to send data packets without collisions. This is achieved by requiring that each station that wishes to send one or multiple data packets must first send a request-to-send packet (RTS) to an intended destination and receive a clear-to-send packet (CTS) from it. In this case the sender uses carrier sensing or packet sensing (i.e., relies on the detection of carrier or the reception of entire error-free packets) to decide when the channel is free. RTSs are required to last a minimum amount of time that is a function of the channel propagation time, and CTSs are required to last longer than an RTS time plus the maximum round-trip time.

A floor acquisition strategy is very attractive in packet-radio networks, because it provides a building block to solve the hidden-terminal problem that arises in CSMA [52]. Variants of this basic strategy can be designed using different types of contention-based MAC protocols like ALOHA or CSMA to transmit RTSs into the channel, and three- or four-way handshakes can be implemented (i.e., RTS-CTS-DATA or RTS-CTS-DATA-ACK). The efficiency of FAMA protocols using carrier sensing to eliminate the hidden-

terminal problems of CSMA has been verified and analyzed [13]. However, the throughput of a FAMA protocol still degrades rapidly once stations start retransmitting unsuccessful RTSs that collide repeatedly with other RTSs. This chapter shows that using tree-splitting algorithms to resolve the collision of RTSs in a FAMA protocol running in a high-speed network improves substantially the performance of the protocol under high-load conditions. The main reason for this is that data packets never collide with control packets in a FAMA protocol and the propagation delays and duration of RTSs and CTSs are much smaller than the duration of data packets. Thus, the average time needed to resolve the collisions of RTSs is very small compared to the duration of data packets. Even when every RTS has to go through collision resolution, the channel can still be used for useful data most of the time.

The chapter is organized as follows. Section 4.1 describes the *Collision Avoidance and Resolution Multiple Access* (CARMA) protocol. This protocol uses non-persistent carrier sensing for the transmission of RTSs and the deterministic tree-splitting algorithm to resolve collisions of RTSs. Many MAC protocols have been proposed based on collision resolution (e.g., RAMA [2], TRAMA [31], DQRUMA [32], DQRAP [57]). The interesting feature of CARMA is the combination of a known collision resolution algorithm with RTS/CTS exchanges, which provides a stable and very efficient protocol that does not require time slotting or availability of base stations capable of detecting multiple simultaneous transmissions. Section 4.2 uses the upper bounds on collision-resolution steps derived in Section 2.3 to compute lower bounds on the throughput achieved by CARMA when a very large population of nodes is assumed. We also show that when the propagation delays and control packets are much smaller than the data packets or packet trains sent by stations, the throughput achieved by CARMA as the arrival rate of RTSs increases is very close to the maximum throughput that can be achieved by any protocol based on collision avoidance (i.e. RTS/CTS exchanges prior to the transmission of data packets). Finally, in Section 4.3 we present simulation results validating the simplifications used in our analytical model, and show that the bounds obtained analytically are close to the throughput values obtained by

simulation.

4.1 Protocol Description

FAMA protocols solve collisions by backing off and rescheduling RTS transmissions [15, 14]. As with CSMA protocols, this procedure yields good results if the RTS traffic is low. However, when the rate of transmissions increases, the probability of collisions also increases and we have a decrease in the system throughput. Eventually, as the RTS transmission rate increases in a FAMA protocol, the constant RTS collisions cause the channel to collapse, bringing the flow of data packets to a halt. One way to stabilize FAMA protocols is, of course, to reduce congestion by increasing the backoff delays. However, this forces nodes to attempt accessing the channel at low rates in order to avoid instability. A well known method to stabilize ALOHA and CSMA protocols is based on collision resolution. In this case, since data packets are used to resolve collisions, the maximum throughput that can be achieved in a stable CSMA or ALOHA protocol is limited [3]. In contrast, CARMA addresses the limitations of FAMA and contention-based protocols using carrier sensing for the transmission of RTSs (with which stations attempt to acquire the floor) and a tree-splitting algorithm to resolve collisions of RTSs. Throughout this chapter, we assume the simple deterministic tree-splitting algorithm for collision resolution introduce in Chapter 2, but more sophisticated algorithms [3] can be used instead. Each station must know the maximum number of stations allowed in the system and the maximum propagation delay in the network.

4.1.1 Basic CARMA

We begin the description of CARMA assuming that all stations are in line of sight of one another. Each station is assigned a unique identifier (ID), a stack, and two variables ($LowID$ and $HiID$). $LowID$ is initially the lowest ID number that is allowed to send an RTS, while $HiID$ is the highest ID number that is allowed to send an RTS. Together, they constitute the allowed ID interval that can send RTSs, i.e., attempt to acquire the floor. If

the ID of a station is not within this interval, the station is not allowed to send its RTS. The stack is the storage mechanism for ID intervals that are waiting to get permission to send an RTS. In CARMA, a station can be in one of the following different states:

- PASSIVE: The station has no local packets pending and no transmissions are detected in the channel.
- RTS: The station is trying to acquire the floor and has sent an RTS.
- XMIT: The station has the floor and is sending data packets.
- REMOTE: The station is receiving transmissions from other stations and has no local packet to send.
- BACKOFF: The station has local packets pending and had to reschedule its request for the floor.

Fig. 4.4 gives a formal description of CARMA. When a passive station has one or multiple packets to send, it first listens to the channel. If the channel is clear (i.e., no carrier is detected) for one maximum round-trip time, the station transmits an RTS. The sender then waits and listens to the channel for one maximum round-trip time, plus the time needed for the destination to send a CTS. When the originator receives the CTS from the destination, it acquires the floor and begins transmitting its data packet burst. The sender is limited to a maximum number of data packets, after which it must release the channel and must compete for the floor at a later time if it still has data packets to send.

If the sender of an RTS does not receive a CTS within a time limit, the sender and all other stations in the system know that a collision has occurred. As soon as the first collision takes place, every station divides the ID interval $(LowID, HiID)$ into two ID intervals. The first ID interval, called the backoff ID interval, is $(LowID, \lceil \frac{HiID+LowID}{2} \rceil - 1)$, while the second ID interval, which is the new allowed ID interval, is $(\lceil \frac{HiID+LowID}{2} \rceil, HiID)$. Each station in the system updates the stack by executing a PUSH-stack command, where the key being pushed is the backoff ID interval. After this is done, the station updates $LowID$ and $HiID$ with the values from the allowed ID interval. This procedure is repeated each time a collision is detected.

Only those stations that were in the RTS state at the time the first collision occurred are allowed into the collision-resolution phase of the protocol. All other stations are in REMOTE state or BACKOFF state, until all collisions are resolved. When all collisions are resolved, those stations in the BACKOFF state have to backoff for a random amount of time. Accordingly, when a passive station obtains a packet to send and the channel is not clear for one maximum round-trip time, the station enters the BACKOFF state.

Collision resolution evolves in terms of collision-resolution intervals. Recall that there are three types of a collision-resolution interval: success, idle and collision. In the first interval of a collision-resolution phase all stations in the allowed ID interval that are in the RTS state try to retransmit an RTS. If none of the stations within this ID interval request the channel, the channel is idle for a time period T_i, i.e. an idle step occurs. At this point, a new update of the stack and of the variables $LowID$ and $HiID$ is due. Each station executes a POP-stack command. This new ID interval now becomes the new $HiID$ and $LowID$. The same procedure takes place if, during the first collision-resolution interval, only one station is requesting the channel, i.e. a success step occurs. In this case, the originator receives the CTS from the destination and begins transmitting its data packet burst, after which the station releases the channel and transitions to the PASSIVE state. The third case of a collision-resolution interval is for multiple stations to request the channel causing a collision step: The stations in the allowed ID interval are once more split into two new ID intervals and the stack as well as the variables for each station are updated. In this case, the duration of the collision-resolution interval is equal to the collision time plus the channel delay.

The algorithm repeats the above three types of collision-resolution steps, until all the RTS collisions have been resolved. As soon as the backoff stack becomes empty and there are no values in the allowed ID interval, all stations know that all the collisions have been resolved. Accordingly, once the tree-splitting algorithm terminates, all stations are either in the PASSIVE state, or in the BACKOFF state. Recall that a station is in the backoff state whenever it has packets to send but, at the beginning of the first collision that started

the tree-splitting algorithm, the station was not in the RTS state. The next access to the channel is driven by the arrival of new packets to the stations and the transmission of RTSs that have been backed off.

To permit the transmission of packet bursts, CARMA enforces waiting periods on receiving stations at strategic points in the operation of the protocol. A station that has received a data packet in the clear must wait for one maximum propagation time after processing a data packet. This allows the sender to send more packets if desired. A station that has understood any control packet must wait for twice the duration of the maximum propagation time. This allows correct RTS/CTS exchanges to take place. On the other hand, if a transmitting station is in the RTS state, the protocol enforces a waiting period of two maximum propagation times after sending its RTS. This allows the destination to receive the RTS and transmit the corresponding CTS. A sending station must also wait one maximum propagation time after the last data packet of its packet train.

4.1.2 Example of CARMA's Operation

As mentioned in Chapter 2, in the deterministic tree-splitting algorithm each station has a distinct position in the leaves of a binary tree based on its ID. If n is the total number of stations in the system, then the corresponding binary tree has $2n+1$ nodes. The root of the tree is labeled n_r and its right and left child n_1 and n_0 respectively. For the remaining nodes, the labels are composed of the parent label, concatenated by a 0 if it is the left child or by a 1 if it is a right child.

For a system with four stations labeled n_{00}, n_{01}, n_{10}, and n_{11}, the corresponding binary tree has seven nodes with the four stations as its leaves. The root is labeled n_r. Its left child is labeled n_1 and is the parent node of the leaves labeled n_{10} and n_{11}, while its right child is labeled n_0 and is the parent node of n_{01} and n_{00}. The binary tree is shown in Fig. 4.1.

Given a tree T, let T_{label} be the subtree at node n_{label}. In our example, the subtree at node n_0 is T_0 and the subtree at node n_1 is T_1.

We now illustrate CARMA's operations using the simple example shown in Fig. 4.1. Initially, the backoff stack is empty and the allowed ID interval contains all the stations in the network, i.e. the allowed ID interval is (n_{00}, n_{11}). The transmission periods for each of the stations and the channel are illustrated in Fig. 4.1.

Assume that, at time t_o, we are at node n_r and we are allowed to listen simultaneously at all the stations of its subtree T_r for a time period of τ seconds. Only one of the following three cases can occur:

- Case 1–Idle: There are no RTSs in any of the leaves (stations) in subtree T_r; therefore, the channel is idle. This lasts an idle transmission period T_i.
- Case 2–Success: There is only one RTS in the subtree T_r; therefore, there is no collision and a station acquires the floor. This lasts one successful transmission period T_s.
- Case 3–Collision: There are two or more stations (leaves) in the subtree T_r sending an RTS; therefore, a collision occurs. This lasts one failed transmission period T_f.

At time t_0, the first collision occurs (Step 1 in Fig. 4.1) with station n_{00} and n_{01} each sending an RTS, while station n_{10} and station n_{11} do not request the channel. All stations detect the beginning of the collision-resolution algorithm and update their stacks and their $LowID$ as well as their $HiID$ values. Stations n_{00} and n_{01} are members of the backoff ID interval and thus they must wait until the collisions in the allowed ID interval are resolved. They both are excluded from sending RTSs. After a time period T_f, stations n_{10} and n_{11} are allowed to request the channel. Since stations n_{10} and n_{11} are in tree T_1 and do not wish to use the channel, an idle period occurs (Step 2 in Fig. 4.1). After $T_i = 2\tau$ seconds, all stations detect that the channel is idle, i.e. that there were no collisions, and they all update their intervals and their stacks. They execute a POP-stack command and the new allowed interval is (n_{00}, n_{01}). T_0 can now proceed to solve its RTS collisions. Both stations n_{00} and n_{01} transmit an RTS control packet (Step 3 in Fig. 4.1) and another collision occurs. Because a collision occurred, the allowed ID interval is split, i.e., the subtree T_0 is split into T_{00} and T_{01}. Station n_{01} is within the allowed interval while station n_{00} must wait since its ID number is not within the allowed ID interval. Since T_{01} has only one station requesting

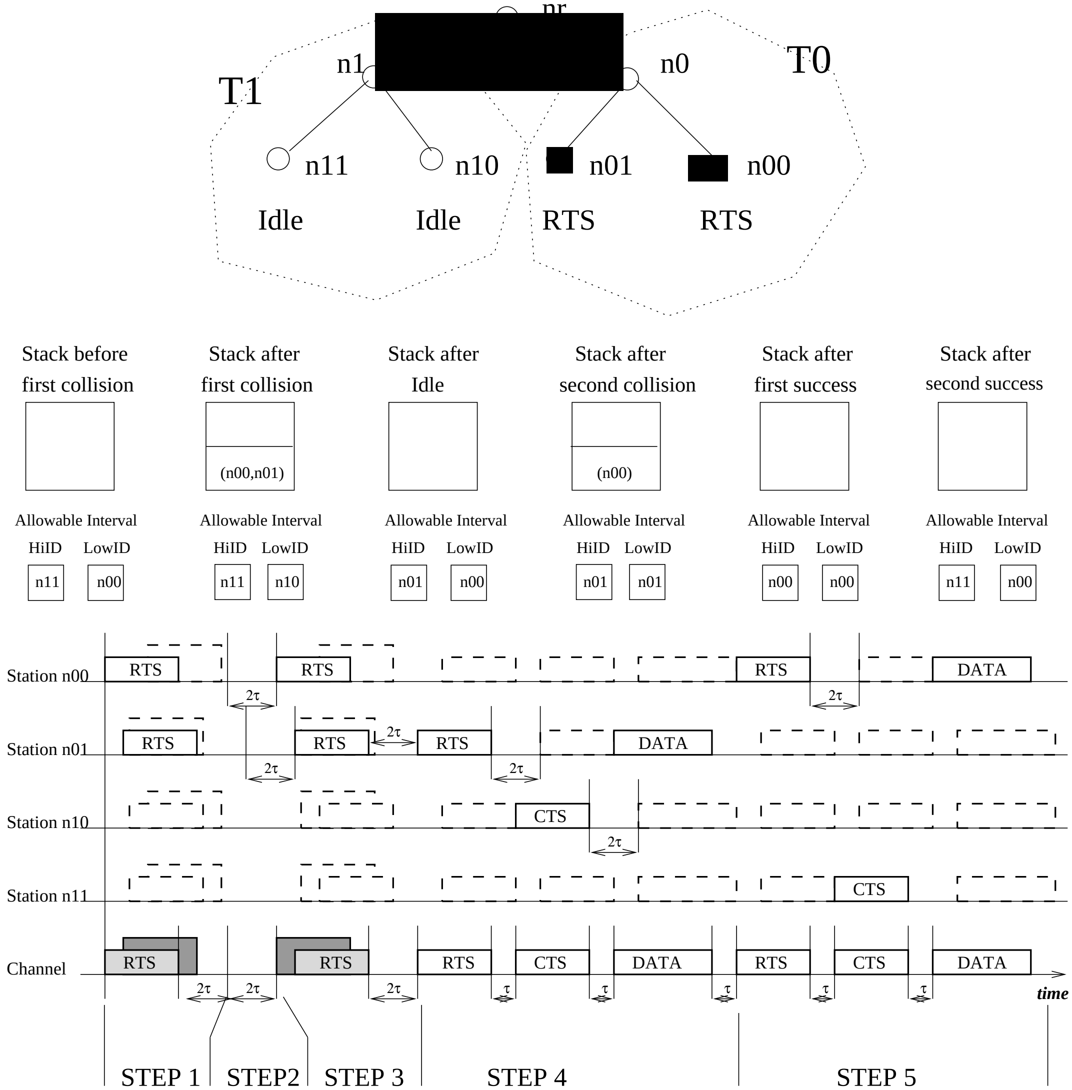

Figure 4.1: Transmission period and tree structure to solve the collisions for a system with $n = 4$ stations out of which $m = 2$ are requesting the floor. $\mathcal{C}(4,2) = 2$, $\mathcal{S}(4,2) = 1$ and $\mathcal{Z}(4,2) = 1$.

the channel, the station can now acquire the floor and transmits its data package (Step 4 in Fig. 4.1). After T_s seconds the stations in CARMA execute a POP-stack command which updates the allowed ID interval. Station n_{00} can now request and acquire the channel transmitting its data packet (Step 5 in Fig. 4.1). Finally, all the stations empty their stacks and update the allowed ID interval permitting all stations to contend in the next round.

Observe that a collision-resolution phase terminates when the stack and the allowed ID interval are both empty.

4.1.3 Handling Hidden Terminals

Our description of CARMA thus far has assumed that all stations can listen to one another. In this section we summarize a simple modification of the basic scheme intended for wireless LANs (WLANs) in which some stations may be hidden from others.

Handling hidden terminals in CARMA is more difficult than in other MAC protocols based on collision avoidance (e.g., MACAW [4], MACA [30], FAMA [16]) because all stations whose transmissions can impact one another must agree on the same state of the collision resolution algorithm, rather than just agreeing on allowing a given sender to transmit a packet without interference.

Fullmer and Garcia-Luna-Aceves [13] have demonstrated sufficient conditions for MAC protocols based on RTS/CTS exchanges to eliminate unwanted collisions due to hidden terminals. For the case of MAC protocols that use carrier sensing, a CTS must be longer than the length of an RTS, plus a maximum round-trip time and processing delays (including radio turn-around times). The net effect of the CTS length suggested in [13] is that of a single-channel. Receiver-based busy tone that forces hidden sources to back off after they are forced to listen to at least part of the CTS sent by a receiver that has accepted the RTS from a given sender. Since the CTS can be only partially overheard by a station after it sends a failed RTS, the CTS cannot be used to convey queue and channel state information.

Consider the case of a WLAN in which stations are trying to communicate with one another over a single hop, or with a base station to reach hidden nodes. The base station is in line of sight of every other station. In this case, each sender is in line of sight of the intended receiver (another station or the base station) and all stations should agree on the same state of a single transmission queue and the same state of the channel. For CARMA to work correctly, the base station must send a second control packet, called the *collision resolution state* packet (CRS), immediately after it sends or hears a CTS, and immediately

after detecting a collision of RTSs. The base station remains silent during idle collision-resolutions steps. The CRS contains the information on the state of the collision-resolution algorithm as perceived by the receiver. The CRS specifies whether or not the collision-resolution step was successful, and in case of a successful collision-resolution step, the CRS indicates the identifier of the successful sender of the RTS.

4.2 Approximate Throughput in CARMA

In Chapter 2, we have computed the average number of steps required to resolve m collisions using the deterministic tree-splitting algorithm. We have also derived upper bounds on the number of collision, idle and success steps (see Section 2.4, Theorem 6 and Theorem 8) required to resolve m collisions in a system with n stations. Notice that the same bounds apply to CARMA with the exception that the steps in CARMA have different length.

The analysis in this section makes the same assumptions introduced in Chapter 2 and uses the same traffic model used for the FAMA-NTR protocol [16]. Given that the upper bounds on the average number of steps (success steps, collision steps, and idle steps) in a collision-resolution period are independent of the number of stations, we approximate the traffic into the channel, with an infinite number of stations, each having at most one RTS to send at any time, and forming a Poisson source sending RTSs with an aggregate mean generation rate of λ packets per unit time. With this model, the average number of RTS arrivals in a time interval of length T is $m = \lambda T$. All data blocks have a duration of δ seconds. The RTS and CTS packets are the same size with a duration of γ seconds. It is not difficult to see that the average channel throughput is given by

$$S = \frac{\overline{U}}{\overline{B} + \overline{I}} \tag{4.1}$$

where $\overline{U}$ is the average utilization time of the channel, during which the channel is being used to transmit data packets, $\overline{B}$ is the expected duration of a busy period, during which

the channel is busy with successful or tree-transmission periods and $\overline{I}$ is the average idle period, i.e., the average interval between two consecutive busy periods.

4.2.1 Unslotted CARMA

Theorem 12: *Let S be the throughput of unslotted CARMA. Then*

$$S \geq \frac{\delta\lambda e^{-\lambda\tau}(\lambda\tau - 1) - \delta\lambda^2\tau}{A \cdot e^{-\lambda\tau} + B} \tag{4.2}$$

where

$$\begin{aligned} A &= -\lambda\delta - 3\lambda\gamma - 5\lambda\tau + \lambda^2\tau\delta + 3.433\gamma\lambda^2\tau + 6.732\tau^2\lambda^2 \\ B &= -6.732\lambda^2\tau^2 - 3.433\gamma\lambda^2\tau - 1 - \lambda^2\tau\delta + \gamma\lambda \end{aligned}$$

Proof: A successful transmission consists of an RTS with one propagation delay to the intended recipient, a CTS with a propagation delay to the sender, and a data packet followed by a propagation delay. Therefore, the average duration period for a successful transmission is

$$T_s = 2\gamma + 3\tau + \delta \tag{4.3}$$

An RTS is said to be successful if during its transmission it is the only packet in the channel. Since across the channel there is a delay of τ seconds before all the other stations in the network detect the carrier signal, the probability of success is equal to the probability that no arrivals occur in τ seconds. After this vulnerability period of τ seconds, all stations defer their transmissions. Given that arrivals of RTSs to the channel are Poisson distributed with parameter λ, we obtain that the probability of success equals $P_s = e^{-\lambda\tau}$.

The number of stations that participate in the collision-resolution phase are $m = \lambda\tau$. The three cases of a collision resolution phase discussed in the previous sections are present within the tree. Each one of them has an average upper bound on the number of steps that depends on the number m of stations requesting the channel but that is independent of the

number n of total stations in the system. In the case of a colliding transmission, i.e. $m > 1$, the time period consists of one RTS package followed by one or more RTSs transmitted by other stations within time Y where $0 \leq Y \leq \tau$, plus one propagation delay τ. It is not difficult to see that the average failed transmission period is bounded by [16]:

$$T_f \leq \gamma + 2\tau \tag{4.4}$$

A waiting period of 2τ seconds is required after a successful transmission or a tree transmission. A busy period is composed of both the successful and the tree transmission periods. The duration of an average busy period equals the sum of the percentage of successful transmission periods times their duration, T_s, plus the percentage of the tree periods times their duration. The tree periods are composed of three parts, corresponding to success, idle, and collision periods, each with a distinct cost and duration. According to the upper bounds derived in Section 2.4, the average busy period can be bounded as follows:

$$\overline{B} \leq T_s \cdot P_s + (T_s m + T_f(1.433 * m - 1) + T_i 0.433 * m)\,(1 - P_s) \tag{4.5}$$

Substituting the values for P_s, T_f, T_s, T_i and m, we obtain

$$\begin{aligned} \overline{B} \quad \leq \quad & \left(\delta + 3\gamma + 5\tau - \lambda\tau\delta - 3.433\lambda\tau\gamma - 6.732\tau^2\lambda\right) \cdot e^{-\lambda\tau} + \\ & \left(\lambda\tau\delta + 3.433\lambda\tau\gamma + 6.732\tau^2\lambda - \gamma - 2\tau\right) \end{aligned} \tag{4.6}$$

The channel carries user data for δ seconds each time an RTS is sent successfully without collision resolution, and δ seconds for each of the RTSs that collide when collision resolution is applied. Therefore, the average utilization is:

$$\overline{U} = \delta \cdot P_s + \delta m(1 - P_s) = (1 - \lambda\tau)\delta \cdot e^{-\lambda\tau} + \delta\lambda\tau \tag{4.7}$$

The average idle period is equal to the average inter-arrival time plus the average waiting period enforced. Thus, $\overline{I} \leq \frac{1}{\lambda} + 2\tau$.

Eq. (4.2) then follows by substituting Eqs. (4.5), (4.7) and $\overline{I}$ into Eq. (4.1). ∎

4.2.2 Slotted CARMA

Theorem 13: *Let S be the throughput of slotted CARMA. Then*

$$S \geq \frac{\delta\lambda^2\tau^2 e^{-\lambda\tau} - \delta\lambda\tau}{A \cdot e^{-\lambda\tau} + B} \tag{4.8}$$

where

$$\begin{aligned} A &= \tau + 0.433\lambda\tau\gamma + 1.299\lambda\tau^2 + \lambda^2\tau^2\delta + 3.433\lambda^2\tau^2\gamma + 5.299\lambda^2\tau^3 - \gamma \\ B &= -2.0\tau - \lambda\tau\delta - 3.433\lambda\tau\gamma - 5.299\lambda\tau^2 + \gamma \end{aligned}$$

Proof: The same assumptions used for unslotted CARMA can be used for slotted CARMA. The channel is slotted and each slot lasts as long as the maximum propagation delay τ. With slotting, stations are restricted to start transmissions only on slot boundaries.

As it was the case in unslotted CARMA, the average duration period for a successful transmission consists of an RTS with one propagation delay to the intended recipient, a CTS with a propagation delay to the sender, and a data packet followed by a propagation delay. Therefore, the average duration period for a successful transmission is $T_s = \delta + 2\gamma + 3\tau$. The probability that an RTS is successful is

$$P_s = P\{k = 1 \text{ arrival in a slot} | \text{some arrivals in a slot}\} = \frac{\lambda\tau \cdot e^{-\lambda\tau}}{1 - e^{-\lambda\tau}} \tag{4.9}$$

In the case of a colliding transmission, i.e. $m > 1$, the time period consists of one RTS followed by a propagation delay τ. All colliding RTSs are sent at the beginning of the same slot. Therefore, we have

$$T_f = \gamma + \tau \tag{4.10}$$

As it was done for unslotted CARMA, $\overline{B}$ can be bounded according to Eq. (4.5). Substituting the values for P_s, T_f, T_s, T_i and m, we obtain

$$\begin{aligned}\overline{B} &\leq T_s \cdot P_s + (T_s m + T_f(1.433m-1) + T_i 0.433m)(1-P_s) \\ &= \frac{(-0.433\lambda\tau\gamma - 1.299\lambda\tau^2 - \lambda^2\tau^2\delta - 3.433\lambda^2\tau^2\gamma - 5.299\lambda^2\tau^3 + \gamma + \tau)e^{-\lambda\tau}}{(1-e^{-\lambda\tau})} + \\ &\quad \frac{(\lambda\tau\delta + 3.433\lambda\tau\gamma + 5.299\lambda\tau^2 - \gamma - \tau)}{1-e^{-\lambda\tau}}\end{aligned} \tag{4.11}$$

For the average utilization we have

$$\overline{U} = \delta \cdot P_s + \delta m(1-P_s) = \frac{\delta\lambda\tau(1-\lambda\tau\cdot e^{-\lambda\tau})}{(1-e^{-\lambda\tau})} \tag{4.12}$$

Finally, the average idle period is the same as in the slotted FAMA-NTR protocol [16] with the only exception that we require a waiting period of 2τ after each collision-resolution period, i.e.

$$\overline{I} \leq \tau \cdot \frac{1}{1-e^{-\lambda\tau}} + 2\tau \tag{4.13}$$

Eq. (4.8), then immediately follows by substituting Eqs. (4.11), (4.12) and (4.13) into Eq. (4.1). ∎

4.3 Numerical Results

In this section we compare CARMA with FAMA-NTR for the cases of a low-speed network (9600 bps) and high-speed network (1Mbps) in which either small data packets (53 bytes) or large data packets (400 bytes) are transmitted. We assume the distance between stations to be the same and define the diameter of the network to be 1 mile. Assuming these parameters, the propagation delay of the channel is 5.4μs. In order to accommodate the use of IP addresses for destination and source, the minimum size of an RTSs and a CTSs is 20 bytes. We normalize the throughput results by defining the following variables

Protocol	*Unslotted Version*	*Slotted Version*
FAMA [16]	$S \geq \frac{ae^{-G}}{(a+b+1)e^{-G}+b+4+\frac{1}{G}}$	$S \geq \frac{aGe^{-G}}{(aG+bG+2G-b-3)e^{-G}+b+4}$
CARMA	$S \geq \frac{(e^{-G}(G-1)-G)a}{A'\cdot e^{-G}+B'}$; $A'=(a+3.433b+6.732)G-a-3b-5$ $B'=(-a-3.433b-6.732)G-\frac{1}{G}+b$	$S \geq \frac{(G\cdot e^{-G}-1)aG}{A'\cdot e^{-G}+B'}$; $A'=(a+3.433b+5.299)G^2+(0.433b+1.299)G+1-b$ $B'=(-a-3.433b-5.299)G+b-2$

Table 4.1: Throughput equations for FAMA-NTR and CARMA protocols

$$\begin{aligned} a &= \frac{\delta}{\tau} && \text{(normalized data packets)} \\ b &= \frac{\gamma}{\tau} && \text{(normalized control packets)} \\ G &= \lambda\tau && \text{(normalized offered load)} \end{aligned} \tag{4.14}$$

Substituting the new normalized variables from Eq. (4.14) into Eqs. (4.2) and (4.8) respectively, we list in Table 4.1 the normalized throughput for each protocol for both the slotted and the unslotted versions. Our results on FAMA throughput are in agreement with the results presented in [16] with the only exception that now $\overline{I}$ and T_f are bounded as in CARMA. Table 4.2 summarizes the protocol parameters used in our comparison.

Network Speed	Packet Size	δ	$a = \frac{\delta}{\tau}$	$b = \frac{\gamma}{\tau}$
9600 bps	424 bits	44166.7 μs	8179.0	3086.4
9600 bps	3200 bits	333333.3 μs	61728.4	3086.4
1 Mbps	424 bits	424 μs	78.5	29.6
1 Mbps	3200 bits	3200 μs	592.6	29.6

Table 4.2: Protocol variables for low-speed networks (9600 bps) and high-speed networks (1 Mbps) with two types of data packets, small (424 bits) or large (3200 bits). The channel delay $\tau = 5.4\mu$s, while the control packets are 160 bits long.

Figs. 4.2 and 4.3 show the throughput S versus the offered load G for CARMA and FAMA-NTR. It is clear that in CARMA slotting does not provide a significant improvement in the performance as compared to unslotted. To achieve high throughput, the size of the control packets need to be small compared to the length of the data packets or packet trains. CARMA behaves like FAMA-NTR when the offered load is small. As the offered load increases, the throughput of FAMA-NTR decreases rapidly, while the throughput of CARMA reaches a constant throughput value. This value is obtained by taking the limit of S as λ goes to infinity. For unslotted CARMA we have

$$\lim_{\lambda\to\infty} S = \lim_{G\to\infty} S = \frac{a}{a + 3.433b + 6.732} = \frac{\delta}{\delta + 3.433\gamma + 6.732\tau} \tag{4.15}$$

while for the slotted CARMA we obtain

$$\lim_{\lambda\to\infty} S = \lim_{G\to\infty} S = \frac{a}{a + 3.433b + 5.295} = \frac{\delta}{\delta + 3.433\gamma + 5.295\tau} \tag{4.16}$$

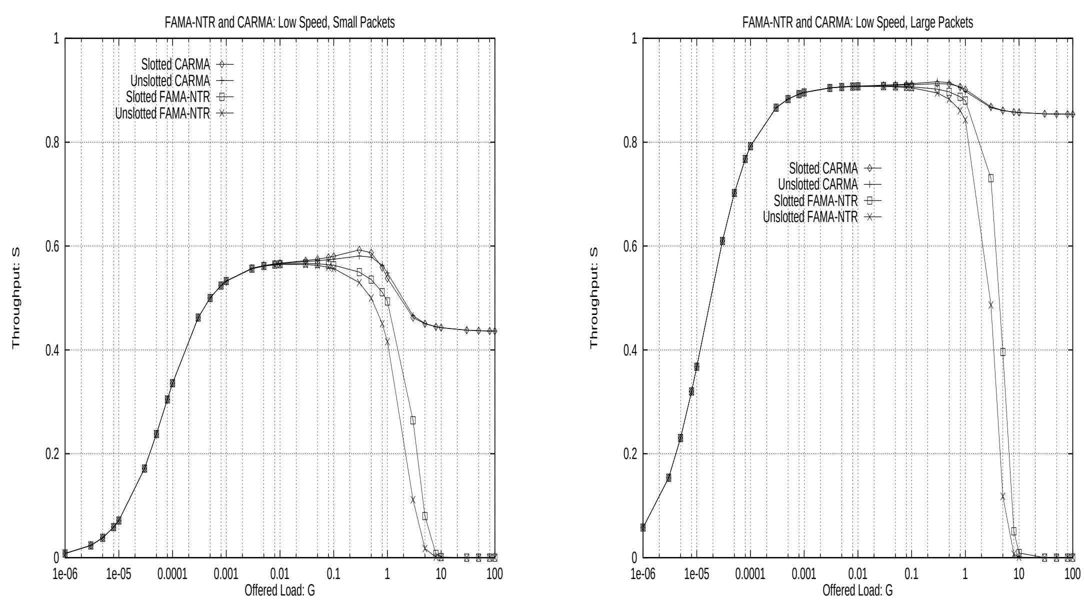

Figure 4.2: Throughput of FAMA-NTR and CARMA for low-speed network.

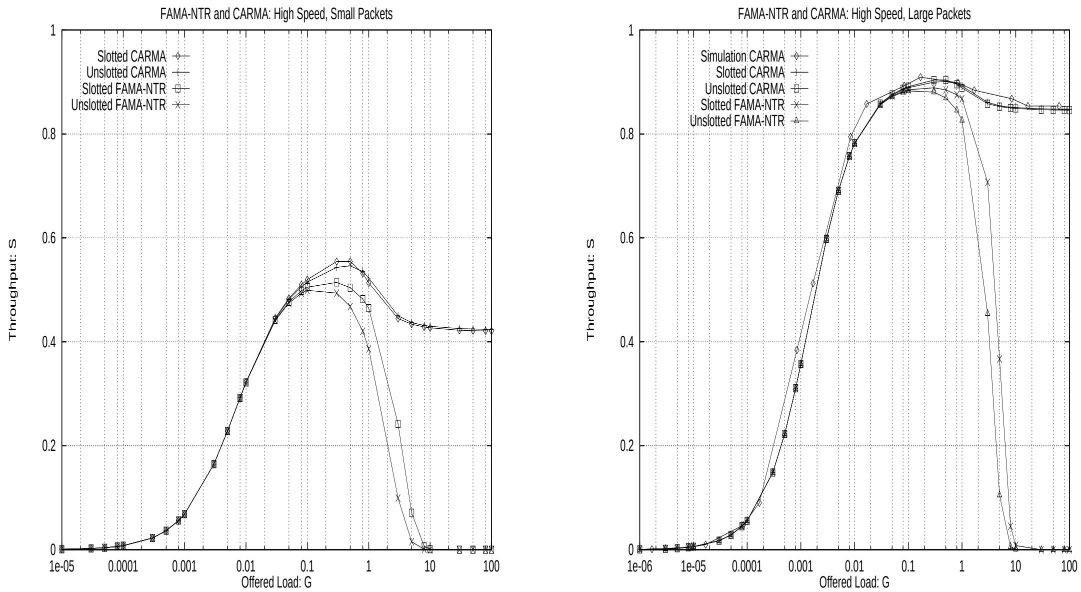

Figure 4.3: Throughput of FAMA-NTR and CARMA for high-speed network.

For slotted as well as for unslotted FAMA-NTR, $\lim_{\lambda\to\infty} S_{fama} = 0$. If we were to have a perfect FAMA protocol in which no collisions of RTSs ever occurred, and a constant flow of data packets, the best possible throughput would be

$$S_{max} = \frac{\delta}{\delta + 2\gamma + 3\tau} \tag{4.17}$$

The ratio of $\lim_{\lambda\to\infty} S$ to S_{max} for unslotted CARMA is then

$$1 \geq \frac{S}{S_{max}} \geq \frac{\delta + 2\gamma + 3\tau}{\delta + 3.433\gamma + 6.732\tau} \tag{4.18}$$

Since $\tau \ll \gamma \ll \delta$, the above result is very encouraging. The result also indicates an improvement if the parameter b is small compared to a. In practice, this effect can be achieved by allowing a station to transmit longer data packets or multiple packets per floor acquisition.

To verify that the throughput values of CARMA approximated by using an infinite population and the upper bounds on the average number of steps required for collision resolution times, provides a good lower bound for any traffic load, we have simulated

slotted CARMA using 100 stations that generate RTSs according to a Poisson probability distribution function. To insure convergence the simulations were done ten times for each given $m = \tau\lambda$ value and allowed to ran for approximately one hour. The results of the simulation are shown in Fig. 4.3 only for the case of long data packets in a high-speed network. These results show that our analysis provides a very good approximation of the throughput.

4.4 Summary

CARMA implements a three-way handshake based on small control packets between sender and receiver, plus a deterministic collision resolution algorithm ensuring that there is always a successful RTS during each busy period. Our simulation validates the simplifying assumptions that we made in order to obtain a lower bound on the throughput as a function of the channel load. Our analysis shows that collision resolution significantly improves the performance of FAMA protocols. The main reason is that for a collision-resolution algorithm the average time required to resolve the collisions of RTSs is much smaller than the time used to transmit the associated data packet trains, which are sent with no collisions due to floor acquisition. We have also shown that, as the arrival rate of RTSs increases, the throughput achieved by CARMA is close to the maximum throughput that any FAMA protocol can achieve if propagation delays and the control packets used to acquire the floor are much smaller than the data packet trains sent by stations. Our analysis of the average number of collisions and idle steps needed to resolve m collisions in a net of n nodes extends prior work on collision resolution, which focused on the total average number of steps needed [9]. These results are essential for obtaining an accurate analysis of MAC protocols in which control packets used to resolve collisions have different length than data packets.

```
Variable Definitions
    CD = Carrier Detected
    T_PROP = Maximum channel propagation delay
    T_PROC = Processing time for carrier detection
    Burst = Number of packets to send in a burst
    Stack = Stack to keep tract of backoffs
    N = Total number of stations
    ID = Station ID number
    LowID = Lowest ID number allow to send RTS
    HiID = Highest ID number allow to send RTS
Procedure START()
Begin
    LowID ⟵ 1
    HiID ⟵ N
    Stack ⟵ NULL
    Timer ⟵ 2 × T_PROP
    While(¬CD ∧ Timer not expired) wait
    If (CD) Then call REMOTE(2 × T_PROP + T_PROC)
    Else call PASSIVE()
End
Procedure PASSIVE()
Begin
    While(¬CD ∧ No Local Packet) wait
    If (CD) Then call REMOTE(2 × T_PROP + T_PROC)
    Else Begin
        If (LowID ≤ ID ≤ HiID) Then
            call RTS(2 × T_PROP + T_TR + T_PROC)
        Else call BACKOFF()
    End
End
Procedure RTS(T_σ i)
Begin
    Transmit RTS Packet
    Timer ⟵ T_σ
    While(¬CD ∧ Timer not expired) wait
    If (Timer expired)
        call Update(LowID, HiID, Stack)
        If (LowID ≤ ID ≤ HiID) Then
            call RTS(2 × T_PROP + T_TR + T_PROC)
        Else call BACKOFF()
    Else
        Receive Packet
        DO CASE of (received packet type)
        Begin
            CTS: call XMIT()
            Error:
                call Update(LowID, HiID, Stack)
                If (LowID ≤ ID ≤ HiID) Then
                    call RTS(2 × T_PROP + T_TR + T_PROC)
                Else call BACKOFF()
        End
    End
End
Procedure BACKOFF()
Begin
    Timer ⟵ T_RTS
    While(¬CD ∧ Timer not expired) wait
    If (CD) Then call REMOTE(2 × T_PROP + T_PROC)
    Else Begin
        If (LowID ≤ ID ≤ HiID) Then
            call RTS(2 × T_PROP + T_TR + T_PROC)
        Else call BACKOFF()
    End
End
Procedure XMIT()
Begin
    Burst ⟵ maximum burst
    While ((Burst > 0) ∧ Local Packet)
    Do Begin
        Transmit Data Packet
        Burst ⟵ Burst - 1
    End
    Timer ⟵ T_PROP
    While (Timer not expired) wait
    call Update(LowID, HiID, Stack)
    If (Local Packet)
        If (LowID ≤ ID ≤ HiID) Then
            call RTS(2 × T_PROP + T_TR + T_PROC)
        Else call BACKOFF()
    Else call PASSIVE()
End
Procedure REMOTE(T_σ)
Begin
    Timer ⟵ T_σ
    While(¬CD ∧ Timer not expired) wait
    If (Timer expired)
        call Update(LowID, HiID, Stack)
        If (Local Packet)
            If (LowID ≤ ID ≤ HiID) Then
                call RTS(2 × T_PROP + T_TR + T_PROC)
            Else call BACKOFF()
        End
        Else call PASSIVE()
    End
    Else Begin
        Receive Packet
        DO CASE of (received packet type)
        Begin
            RTS: call REMOTE(2 × T_PROP + T_PROC)
            CTS:
                if (Destination ID = Local ID) Then
                    Transmit CTS Packet
                call REMOTE(2 × T_PROP + T_PROC)
            DATA:
                if (Destination ID = Local ID)
                Then pass packet to upper layer
                call REMOTE(T_PROP + T_PROC)
            ERROR: call REMOTE(2 × T_PROP + T_PROC)
        End
    End
End
```

Figure 4.4: CARMA Specification

5. CARMA-NTQ

Several stable MAC protocols have been proposed in the past based on tree-splitting algorithms (e.g., [9, 17, 40]). Those protocols in which data packets are used to resolve collisions achieve throughput below 0.6 [55]. More recent MAC protocols have been proposed that implement collision resolution using either control packets that are much smaller than data packets, or are based on the ability of the transmitter to abort transmission rapidly after detecting collision (e.g., [6, 18, 31]). Among those stable MAC protocols that achieve high throughput, some build a separate queue for the transmission of data packets, in addition to the stack or queue of the control packets used for collision resolution. However, prior protocols based on data transmission queues and collision resolution require the establishment of time slots or mini-slots for the transmission of control packets [53, 57].

The concept of "Group Allocation Multiple Access" (GAMA) was first introduced by Muir and Garcia-Luna-Aceves [41] to provide performance guarantees in asynchronous MAC protocols. A GAMA protocol dynamically divides the channel into cycles of variable length; each cycle consists of a contention period and a queue-transmission period. During the contention period, a station with a message to send competes for the right to be added to the transmission queue; this is done using a request-to-send/clear-to-send (RTS/CTS) message exchange with carrier sensing. The queue-transmission period is a variable-length train of packets, which are transmitted by stations that have been added to the transmission queue by successfully competing for it. As long as a station maintains its position in a transmission queue, it transmits packets without collisions. Variants of this basic strategy can be designed using different types of contention-based MAC protocols like ALOHA or CSMA to transmit RTS packets into the channel.

We have shown in Chapter 4 that collision resolution when applied to the RTS/CTS handshake used in many MAC protocols can improve the throughput of the system substantially. In this chapter we introduce a stable multiple access protocol for broadcast channels shared by bursty stations, which we call CARMA-NTQ (for collision avoidance and resolu-

tion multiple access with non-persistence and transmission queues). Like previous efficient MAC protocols based on tree-splitting algorithms (e.g., DQRAP), CARMA-NTQ maintains a distributed queue for the transmission of data packets and a stack for the transmission of control packets used in collision resolution. However, CARMA-NTQ does not require the mini-slots commonly used in protocols based on collision resolution. CARMA-NTQ dynamically divides the channel into cycles of variable length; each cycle consists of a contention period and a queue-transmission period. The queue-transmission period is a variable-length train of packets, which are transmitted by stations that have been added to the distributed transmission queue by successfully completing a collision-resolution round in a previous contention period. During the contention period, stations with packets to send compete for the right to be added to the data-transmission queue using a deterministic first-success tree-splitting algorithm, so that a new station is added to the transmission queue.

CARMA-NTQ provides dynamic reservations of the channel, together with collision resolution of the reservations requests based on the deterministic tree-splitting algorithm introduced in Chapter 2. Like GAMA [41], CARMA-NTQ builds a dynamically-sized cycle that grows and shrinks depending upon traffic demand. Each cycle consists of a contention period and a queue-transmission period during which one or more stations transmit data packets without collision. A position in the transmission queue is allocated to an individual station during the contention period, and a station can continue to transmit in this position as long as it has data to send to any other station in the network. Stations compete to acquire the right to be in the transmission queue based on the deterministic tree-splitting algorithm.

A station attempts to join the transmission queue during contention intervals by sending a request-to-send (RTS)to any intended receiver, who sends a clear-to-send (CTS) if the station can join the queue. RTSs are sent according to a deterministic tree-splitting algorithm that resolves one request for the queue from all those that arrive during the same contention interval (a single successful RTS/CTS exchange is allowed). Access time to the channel is divided into rounds of transmissions for all members of the transmission queue,

which we call a queue-transmission period, followed by short contention periods during which stations attempt to join the queue. The queue-transmission period is a variable-length train of packets from stations that have been added to the transmission queue by successfully completing a collision-resolution step in a previous contention period. The control packets used in each contention period are much smaller than data packets.

The chapter is organized as follows. Section 5.1 describes CARMA-NTQ in detail. For simplicity, we present CARMA-NTQ assuming a fully connected network in which collision resolution is implemented using an RTS/CTS message exchange with non-persistent carrier sensing. CARMA-NTQ is more attractive than previous dynamic reservation schemes for wireless (and wired) LANs [31, 34, 45, 48, 53, 55] in that it does not require time synchronization or the definition of control frames of fixed duration over which the slots for the data frame can be reserved. It is also more attractive than token passing schemes in that no fixed schedule exists for passing the token. Section 5.1 also outlines approaches to apply CARMA-NTQ to wireless LANs with hidden terminals.

When m stations request to be added to the transmission queue during the same contention period, their requests collide in the channel, and the tree-splitting algorithm used in CARMA-NTQ incurs a number of additional collisions, idle times, and one successful RTS/CTS exchange in resolving such collisions, i.e., in adding a station to the transmission queue. Because no time slotting is used in CARMA-NTQ, a collision, an idle time, and a successful RTS/CTS exchange all last different times and contribute differently to the protocol overhead. Previous results on the number of steps needed for collision resolution in tree-slitting protocols focus on the total number of steps [9, 17, 40], rather than on computing the number of idle steps, collision steps, and success steps (one for first-success).

Using the upper bounds on the number of idle steps and collision steps required by the deterministic tree-splitting algorithm to resolve collisions up to the first successful RTS/CTS exchange derived in Section 2.4, Section 5.2 provides a lower bound for the throughput achieved with CARMA-NTQ as a function of the size of the transmission queue and the number of queue-addition requests that need to be resolved. This bound is based on the

upper bound on the average number of collision resolution steps needed to resolve a given number of queue-add requests. The throughput obtained in CARMA-NTQ is described as a function of the size of the transmission queue and the number of stations that request to be added in the queue in a contention interval. It is shown that, as the average duration of the data transmission queue or the persistence of the station in the data-transmission queue increases, CARMA-NTQ gives every station a place in the transmission queue. Analytical and simulation results in Section 5.3 show that the throughput in CARMA-NTQ approaches the channel capacity as propagation delays decrease. The simulation results show that our lower bound on the throughput is a very good analytical approximation of the protocol's throughput.

5.1 CARMA-NTQ

To simplify our exposition and the analysis of CARMA-NTQ, we assume a fully connected network in which all stations can hear one another. The last part of this section summarizes approaches to apply CARMA-NTQ to wireless LANs with hidden terminals.

5.1.1 Basic Operation

With CARMA-NTQ, the channel access time is divided into cycles, with each such cycle consisting of a small, dynamically-sized contention period and a dynamically-sized queue-transmission period. Stations are assumed to monitor the state of the channel continuously while they are not transmitting.

The queue-transmission period consists of the collision-free transmissions from the stations with a position in the transmission queue. Each station allocated to the transmission queue transmits a packet as soon as the packet from the previous station in the queue is received; accordingly, the maximum spacing between packets is twice the propagation delay. A station wishing to send one or more packets has to acquire a position in the transmission queue before transmitting its data packets.

Stations compete to reserve a position in the transmission queue during the contention periods between queue-transmission periods. During a contention period, stations follow a non-persistent CSMA strategy for the transmission of the RTSs used to request a place in the transmission queue. An RTS can be directed to any other station. The sender of an RTS waits and listens to the channel for one maximum round-trip time, plus the time needed for the destination's CTS to arrive. If the CTS is not corrupted and is received within the time limit, the station is added to the transmission queue, starting with the next queue-transmission period. If the CTS is not received, all stations monitoring the channel detect a collision.

To provide delay guarantees for stations that are already members of the transmission queue, the transmission queue and the duration of each transmission turn in the queue can be allocated maximum lengths. Once the transmission queue reaches its maximum length, stations are not allowed to request to be added to the queue, until at least one member of the queue ends transmitting its packets and leaves the queue.

Although each station transmits an RTS only when it determines that the channel is free during a predefined amount of time in the contention period, collisions of RTSs may still occur due to propagation delays. RTSs are vulnerable to collisions for time periods equal to the propagation delays between the senders of RTSs. CARMA-NTQ uses a deterministic tree-splitting algorithm to resolve collisions of RTSs, which are detected by the absence of a CTS. A step in a contention period can be an idle period, an RTS collision period, or a successful RTS/CTS exchange, and all of these are much shorter than a single data packet.

Once the collision-resolution algorithm starts by the collision of RTSs from a set of stations, other stations who want to be added to the transmission queue must wait for one of the stations who started the current round of the collision resolution algorithm to be added to the transmission queue through a successful RTS/CTS exchange; all stations know when this occurs by means of a stack that is maintained distributedly, which is described subsequently. We assume a non-persistent policy in which stations who are not in the transmission queue and are not involved in an collision resolution must backoff for

a random period of time if they receive local packets to send and need to be added to the transmission queue.

A station that is not transmitting must be listening to the channel continuously and knows the current state of the channel. Based on its knowledge of the state of the transmission queue and the current step of the collision-resolution algorithm used to add stations to the queue, a station can be in one of the following states:

- XMIT: the station is part of the transmission queue.
- RTS: the station is trying to acquire a position in the transmission queue and is participating in the collision-resolution algorithm.
- BACKOFF: the station is waiting for a backoff time to complete as a result of receiving local packets to send outside an access period.
- REMOTE: the station is not part of the transmission queue and knows that a set of collisions of requests for the queue are still being resolved.
- PASSIVE: the station is not part of the transmission queue, has no local packets to send, and knows that no prior requests for the queue are being resolved.

5.1.2 Information Maintained and Exchanged

Each station is assigned a unique identifier, knows the maximum number of stations allowed in the network and the maximum propagation delay, and maintains a stack and two variables ($LowID$ and $HiID$). $LowID$ is initially set to 1 and denotes the lowest ID number that is allowed to send an RTS. $HiID$ is the highest ID number that is allowed to send an RTS; it is initially set to the largest number of stations allowed in the network. $LowID$ and $HiID$ constitute the allowable ID interval that can send RTSs. If the ID of the station is not within this interval or the station has already been assigned to the data transmission queue, it cannot send an RTS. The stack is simply a storage mechanism for ID intervals that are waiting for permission to send an RTS.

Each station also maintains the state of the transmission queue, i.e., it knows the members in the transmission queue and their position in the queue and the beginning

of each queue-transmission period. RTSs and CTSs specify the IDs of the sender and the intended receiver, as well as the allowable ID interval known to the sending station. Including the allowable ID interval in control packets allows listening stations to be updated on the state of the collision resolution. A data packet contains: (a) the user data and destination of the packet; (b) the allowable ID interval known to the sending station; (c) the state of the transmission queue, such as the number of stations in the queue and the position of the sending station in the queue; and (d) the state of the transmission for the sending station, such as the length of the current packet, whether the station is leaving the transmission queue, and the length of the *next* packet if variable length packets are allowed. The information in control and data packets updates listening stations on the state of the channel.

5.1.3 Adding Members to the Transmission Queue

Stations can request to be added to the transmission queue only during contention periods. If the maximum length of the transmission-queue is reached, the contention steps are skipped, until space is made available by one or multiple stations leaving the queue. Stations are allowed to send RTSs to be added to the transmission queue only during the first few seconds of a contention period. This time period is called the *access period* and is used in order to avoid collisions of RTSs with the beginning of the transmission queue. For the purposes of analysis, we assume subsequently that the access period lasts τ seconds, but it is a configurable parameter. A contention period consists of several steps (i.e., a success, and idle or a collision of control packets) of a deterministic tree-splitting algorithm based on the allowable ID interval, which we describe in Chapter 2.

If a PASSIVE station receives local packets to send during an access period, it first listens to the channel. If the channel is clear (i.e., no carrier is detected), the station enters the RTS state by sending an RTS. The sender then waits and listens to the channel for one maximum round-trip time plus the time needed for the destination to send a CTS. When the originator receives the CTS from the destination, it is added to the data transmission

queue and enters the XMIT state; the station can start sending collision-free packets in the following queue-transmission period. We do not specify how stations are ordered in the transmission queue, a simple approach is to add a station at the end of the queue.

On the other hand, if a PASSIVE station obtains local packets to send outside an access period, or receives the packets within an access period but detects carrier, it enters the BACKOFF state. The station then computes a random backoff time and attempts to enter the data transmission queue after that time. Because a station listens to the channel continuously, the backoff time can be applied immediately, i.e., without waiting for any ongoing collision resolution to be completed. The backoff time should be a multiple of the current cycle length. If the backoff time-out of a station expires when there is an ongoing collision resolution, the station backs off again. We can assume that the backoff time is always extended so that a station always comes out of backoff state at the beginning of an access period.

If the sender of an RTS does not receive the corresponding CTS within the allocated time, this station and all other stations in the network assume that a collision has occurred. The collision-resolution algorithm is started with the first round of RTS collisions. As soon as a collision is detected, every station divides the ID interval $(LowID, HiID)$ into two ID intervals. The first ID interval, which we will call the backoff interval, is $(LowID, \lceil \frac{HiID+LowID}{2} \rceil - 1)$, while the second ID interval, the allowable ID interval, is $(\lceil \frac{HiID+LowID}{2} \rceil, HiID)$. Each station in the system updates its stack by executing a PUSH stack command, where the key being pushed is the backoff interval. After this is done, the station updates $LowID$ and $HiID$ with the values from the allowable ID interval and the queue-transmission period follows. This procedure is repeated each time a collision is detected.

Hence, a station knows if collisions are currently being resolved when its local stack is not empty and the allowable ID interval does not equal the entire ID interval. A PASSIVE station that detects a collision of RTSs goes to the REMOTE state. A station in REMOTE state goes to PASSIVE state as soon as it detects that its local stack is empty and the

allowable ID interval is empty. If a station is in REMOTE state and obtains one or more packets to send, it goes immediately to the BACKOFF state.

Only those stations that are in the RTS state participate in the current round of the collision-resolution algorithm to be added to the transmission queue. All stations in the allowable ID interval that are in the RTS state transmit an RTS in the next contention period. Stations outside of the transmission queue that did not start the current round of collision resolution are forced into REMOTE or BACKOFF state.

Each step of the collision-resolution algorithm can be one of the following three cases:

- Case 1–*Idle*: There is no station in the RTS state whose ID is within the allowable ID interval. Therefore, the channel must remain idle one collision-resolution step and no new member is assigned to the transmission queue. An idle step of collision resolution lasts a maximum round-trip time. After that time, the stack and the variables $LowID$ and $HiID$ are updated and each station executes a POP command in the stack. This new ID interval now becomes the new $HiID$ and $LowID$.

- Case 2–*Success*: There is a single station with an RTS to send whose ID lies within the allowable ID interval. In this case, a single station is able to complete an RTS/CTS handshake successfully, enters the XMIT state, and is added to the transmission queue. The duration of a successful collision-resolution step lasts one RTS, one CTS and two channel propagation delays; after that time, the stack and the variables $LowID$ and $HiID$ are updated; each station executes a POP command in the stack. This new ID interval now becomes the new $HiID$ and $LowID$.

- Case 3–*Collision*: There are two or more stations with RTSs to send whose IDs are within the allowable ID interval; therefore, each such station sends an RTS, creating a collision. Because stations use carrier sensing, a collision period can last no more than an RTS plus a maximum round-trip time. The stations in the allowable ID interval are again split into two new ID intervals; the stack and the variables for each station are updated.

The tree-splitting algorithm executes several collision resolution steps (idle, success, or collision) during a contention period, until one request for the queue has been resolved. Again, all stations know when the tree-splitting algorithm terminates its current round, because the first successful RTS/CTS exchange marks the termination of the contention round. Once termination is detected, the allowable ID interval is updated to include all the ID range and the backoff stack is empty. The stations that cannot be added to the transmission queue after the first-success tree-splitting terminates enter the BACKOFF state.

To ensure fairness within the splitting algorithm, the position of the stations in the tree (which is equivalent to changing the ID number) can vary after each round of collision resolution. For example, the ID numbers of the stations can rotate cyclically. Each station increases its ID number by one and the last station takes the ID number of the previous first station.

5.1.4 Using the Transmission Queue

A maximum interval allowed between two consecutive packets in the queue can be defined based on the maximum propagation delay and processing delays. Because stations are at most τ seconds apart from one another, two consecutive packets in the queue cannot be separated by more than 2τ seconds plus processing delays (including the turn around times of the radios), which we call *maximum queue transmission gap* and assume that it lasts ϕ seconds.

A station in the queue must transmit when its turn comes so that its transmission reaches other stations within ϕ seconds of the previous transmission. A "silence turn" could occur in the transmission queue if a station is not ready to transmit a data packet when its turn comes but is not ready to leave the queue. Rather than allowing silence turns to take place, CARMA-NTQ requires a station to transmit either a data packet or a short control packet when its turn comes in the queue. The control packet carries no user data and is simply used to convey an update of the state of the queue and the collision resolution process.

5.1.5 Deleting Members from the Transmission Queue

A deletion from the transmission queue occurs when a station in the XMIT state ends transmitting a packet train and gets out of the queue to remain quiet. In this case, the information contained in a data packet notify the stations when the sending station is sending its last packet before leaving the queue.

Deletions from the transmission queue can also occur due to failures. Failures can be handled by listening to the channel and different approaches can be taken. One approach consists of stations sending a jamming packet after not receiving the data or control packet from the station with the current turn. The duration Γ of the jamming packet is longer than the maximum propagation delay. Because the maximum queue transmission gap lasts ϕ and stations are at most τ seconds apart, every station knows that a short failed turn ends after at most $\phi + \Gamma + 2\tau$ seconds.

A better approach consists of giving jamming priority to the station with the next turn in the queue. The station that follows a failed station in the queue sends its jamming packet after not detecting the start of the prior stations packet for ϕ seconds. All other stations wait for a period of $\phi + 2\tau$ seconds to start receiving either the packet from the current turn or the jamming from the station of the next turn. This increases the failed turn slightly, but reduces considerably the number of transmissions needed, because it is unlikely for two stations to fail with adjacent turns in the queue. After a stations turn is declared as failed, that station must try to rejoin the queue like any other out-of-queue station.

5.1.6 CARMA-NTQ Example

The following example illustrates the way in which stations are added to the transmission queue. Consider the case of a network with four stations and assume that, at time t_0, station n_{11} is the only station assigned to the transmission queue and has five data packets to transmit. The diagram of Fig. 5.1 starts with the beginning of a new cycle, i.e., a contention period. The allowable ID interval includes all the ID range and the backoff stack is empty (step 1).

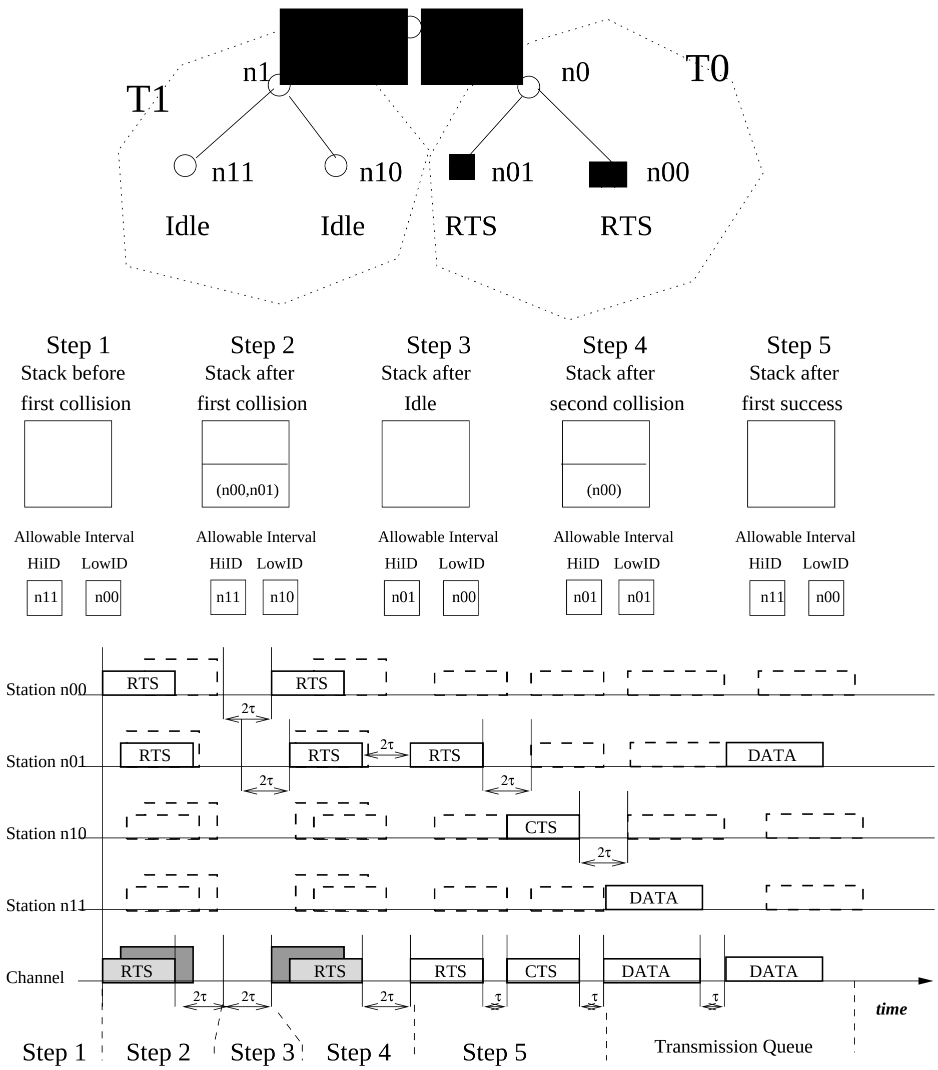

Figure 5.1: Tree structure for an example with $n = 4$ stations and $m = 2$ stations are requesting to be admitted into the transmission queue. The termination criteria is the first successful RTS/CTS exchange.

During the access period of the contention period, packets arrive at stations n_{01} and n_{00}, and a collision occurs with station n_{00} and n_{01} each sending an RTS in the same access period, while station n_{10} and station n_{11} do not request to be added to the transmission queue (see Fig. 5.1). Station n_{11} does not need to send a request, because it is already assigned to the transmission queue, and all other stations know that it still has five data packets to transmit. Let station n_{01} have three data packets to send, while station n_{00} has

only one data packet to transmit. At time t_0 the first collision occurs, all stations including n_{01}, notice the beginning of a new round of the collision-resolution algorithm, as well as the beginning of the contention period. All stations update their stacks and their $LowID$ and $HiID$ values. After at most an RTS plus a maximum round-trip time has elapsed, stations n_{00} and n_{01} backoff and will therefore wait until the collisions in the allowed-ID set are resolved. They both are excluded from sending RTSs. Stations n_{10} and n_{11} are allowed to request the channel. In the next contention period (step 2), stations n_{00} and n_{01} both are on hold, because they must wait until the collisions in the allowable-ID set are resolved. They both are excluded from sending RTSs. Stations n_{10} and n_{11} are allowed to request the channel. Since stations n_{11} and n_{10} do not wish to be added to the transmission queue, an idle contention period occurs (Case 1). After 2τ seconds, all stations notice that the channel is idle, which means that there were no collisions; accordingly, the stations in the system update their intervals and the stack. They execute a POP-stack command and the new allowable interval is (n_{00}, n_{01}) (step 3). Stations n_{00} and n_{01} transmit an RTS control packet and another collision occurs (step 4). Because a collision occurred, the allowable ID interval is split into two halves. In the next contention step (step 5), station n_{01} is within the allowable ID interval, while the n_{00} station must wait; its ID interval is at the top of the stack. Since the next ID interval has only one station sending an RTS, station n_{01} is assigned to the transmission queue after an RTS, a CTS plus a maximum round-trip time. During the successful RTS/CTS exchange, all other stations learn that station n_{01} has three packets to send. This marks the termination of the contention period and the stack as well as the allowable ID interval are updated to the original state at the beginning of the contention period, i.e., the stack is empty and the ID interval is (n_{00}, n_{11}). When contention step 5 ends, station n_{01} begins transmitting its data packet immediately after it has read the packet transmitted by the preceding queue member, which is station n_{11}; after that, station n_{11} has four data packets left while station n_{01} has two packets. Station n_{00} must reschedule its request to a later access period. The duration of the queue-transmission period is the size of the length of two packet plus the two channel delays.

5.1.7 Handling Hidden Terminals in CARMA-NTQ

Our description of CARMA-NTQ thus far has assumed that all stations can listen to one another. In this section we summarize a simple modification of the basic scheme intended for wireless LANs in which some stations may be hidden from others and where some stations may capture one of multiple RTS that collide. Not surprisingly, this modification is much the same as the one presented for CARMA in Chapter 4. We assume that the stations' physical layer adhere to IEEE 802.11 specifications [42].

Consider the case of a wireless local area network (WLAN) in which stations are trying to communicate with one another over a single hop, or with a base station to reach hidden nodes. In this case, each sender is in line of sight of the intended receiver (another station or the base station) and the base station, and all stations should agree on the same state of a single transmission queue and the same state of the channel. For CARMA-NTQ to work correctly in this case, the base station must send the feedback to all senders. This eliminates the possibility of different receivers sending CTS due to capture of one of many RTSs. Because the length of the CTS from the base station lasts long enough to ensure that all stations that send an RTS detect the noise from the end of the CTS. Furthermore, the base station must send a second control packet, which we call the *channel and queue state* packet (CQS) immediately after it sends a CTS. The CQS has no length restrictions relative to an RTS and contains all the information on the state of the queue and channel as perceived by the receiver. The minimum channel information needed in a CQS consists of the ID of the sender of the successful RTS. The base station is the one who should send the CTS and CQS to all other stations, because it is the only station that must hear and be heard by all other stations. To cope with channel errors and fading, the base station also sends a probe packet to each member of the transmission queue. A station in the transmission queue sends its packet only after receiving the probe packet.

5.2 Throughput Analysis

The analysis in this section uses the same traffic model used by Kleinrock and Tobagi [33] to analyze CSMA protocols. It is assumed that there is large number of stations, each forming a Poisson source of RTSs (both new and retransmitted). The aggregate mean generation rate of RTSs by all stations is λ RTSs per unit time. Each station is assumed to have at most one RTS to transmit at any given time. With this model, the average number of RTS arrivals in a time interval τ is $\lambda\tau$, i.e., $m = \lambda\tau$. All data packets have a duration of δ seconds, and the time to transmit a control packet (RTS or CTS) is γ seconds. The number of packets in a message is a random variable, and the probability that a message will complete its transmission (in a given cycle) is given by $q = \frac{1}{N}$ where N is the average number of packets in a message. We also assume that the time to transition between transmit and receive states is negligible.

Fig. 5.1 shows the transmission period for CARMA-NTQ. As Fig. 5.1 illustrates, a CARMA-NTQ busy period is either an idle, a successful or a tree transmission, followed by an idle period. Given that the upper bounds on the average number of collision-resolution steps are independent of the number of stations, we approximate the traffic into the channel with an infinite number of stations, each having at most one RTS to send at any time, and forming a Poisson source sending RTSs with an aggregate mean generation rate of λ packets per unit time. With this model, the average number of RTS arrivals in a time interval of length T is λT, i.e., $m = \lambda T$. To simplify our analysis, the members in the data transmission queue are ordered by the number of packets (in the message) remaining to be transmitted.

Theorem 14: *The throughput of CARMA-NTQ is given by*

$$S \geq \frac{\rho\delta}{Ae^{-\lambda\tau} + B} \tag{5.1}$$

with

$$A = \left[(2\tau + 2\lambda\tau^2 + \gamma + \gamma\lambda\tau)\log(\lambda\tau) + (3\lambda\tau^2 + \gamma\lambda\tau + 3\tau + 3\gamma)\right](\Pi_h - 1)$$

$$B \quad = \quad (2\tau + \gamma)(1 - \Pi_h)\log(\lambda\tau) + (5\tau + 3\gamma)(1 - \Pi_h) + (2 + \rho)\tau + \frac{\Pi_o}{\lambda} + \rho\delta \tag{5.2}$$

Proof of Theorem 14: The average channel throughput is equal to

$$S = \frac{\overline{U}}{\overline{B_c} + \overline{B_q} + \overline{I}} \tag{5.3}$$

where $\overline{U}$ is the average utilization time of the channel, during which the channel is being used to transmit data packets; $\overline{B_c}$ is the expected duration of the contention period; $\overline{B_q}$ is the expected duration of the queue-transmission period; and $\overline{I}$ is the average idle period, i.e., the average interval between two consecutive busy periods.

The length of the average utilization period $\overline{U}$ depends on the average number of members in the transmission queue (ρ) and increases the throughput of the system.

$$\overline{U} = \rho\delta \tag{5.4}$$

The average queue-transmission period is equal to the average number of queue members (ρ) times the transmission period for one packet ($\delta + \tau$); therefore,

$$\overline{B_q} \quad = \quad \rho \cdot (\delta + \tau) \tag{5.5}$$

If the data transmission queue is empty, the length of the idle period is given by the next RTS arrival into the channel plus a waiting period of 2τ. In contrast, when the data transmission queue is not empty, the length of the idle period is limited by the start of the next transmission period which is 2τ seconds. Let Π_0 denote the probability that the data transmission queue is empty, we obtain

$$\overline{I} \quad = \quad \Pi_0\left(\frac{1}{\lambda} + 2\tau\right) + (1 - \Pi_0)2\tau = \frac{1}{\lambda}\Pi_0 + 2\tau \tag{5.6}$$

The probability P_z that a contention period is idle is equal to the probability that no packets arrive during the access period τ, which can be express as

$$P_z = e^{-\lambda\tau} \tag{5.7}$$

For a station to be added to the data transmission queue, the RTS/CTS exchange must be successful, the RTS must be the only one in the channel during its access period, and the data transmission queue must not be full. The probability P_r of this happening equals the probability that only one RTS arrives during the τ seconds of the access period. If there are already h queue members, no new member can be added. In such a case the contention would be skipped, until a space in the data transmission queue is available. Therefore, P_r can be express as

$$P_r = \lambda\tau e^{-\lambda\tau} \tag{5.8}$$

P_r is the fraction of the cycles that have one RTS during the access period, while $1 - P_r - P_z$ is the fraction of the cycles that have more than one RTSs during the access period. The probability P_c that a contention period results in an RTS collision is the same as the probability that more than one message arrives during an access period. This value can be express as:

$$P_c = 1 - \lambda\tau e^{-\lambda\tau} - e^{-\lambda\tau} \tag{5.9}$$

The states of the queue-transmission period can be represented by a Markov chain. Fig. 5.2 is an example for a network that allows at most $h = 4$ stations to be members of the queue-transmission period. Transitions from one state to the other have been adequately labeled, where

$$W = P_r + P_c \tag{5.10}$$

and

$$V = P_z \tag{5.11}$$

W is the probability that a new member is added to the transmission queue per contention period, while V is the probability that it is not added. The states in the Markov chain represent the probability Π_k that k members are in the data transmission queue. Notice that the number of members in the data transmission queue can only increase by one, but can decrease by up to k. At most, one new member can be added to the data transmission queue per contention period, but any number of members can be released from the queue per queue-transmission period.

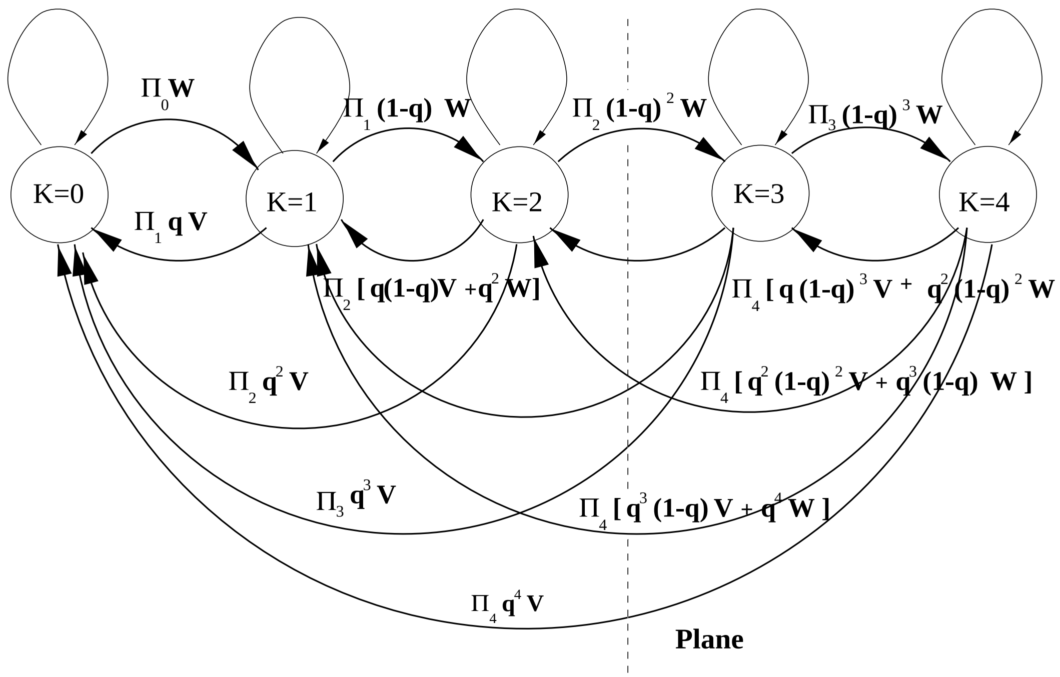

Figure 5.2: Markov Chain defining the transitions from one state to the others. The given example is for a network that allows, up to four members in the data transmission queue. Only a subset of the transition probabilities are shown

We can generalize our example and define h as the maximum number of members in the data transmission queue allowed by the network. If we draw a line between any two consecutive states of the Markov chain, then the flow of the transitions going in one direction

has to be equal to the flow in the other direction. Therefore, for any cut within the Markov chain excluding the first and the last state, i.e., for $1 \le k \le h-2$,

$$\begin{aligned} \Pi_k W(1-q)^k \quad = \quad & \sum_{i=1}^{h-k-1} \Pi_{k+i} \left(W \sum_{j=i+1}^{k+i} q^j (1-q)^{k+i-j} + V \sum_{j=i}^{k+i} q^j (1-q)^{k+i-j} \right) + \\ & \Pi_h \left(W \sum_{j=h-k+1}^{h} q^j (1-q)^{h-j} + V \sum_{j=h-k}^{h} q^j (1-q)^{h-j} \right) \end{aligned} \tag{5.12}$$

If we divide both sides by $W(1-q)^k$, the result is

$$\begin{aligned} \Pi_k \quad = \quad & \sum_{i=1}^{h-k-1} \Pi_{k+i} \left(\sum_{j=i+1}^{k+i} \frac{q^j}{(1-q)^{j-i}} + \frac{V}{W} \sum_{j=i}^{k+i} \frac{q^j}{(1-q)^{j-i}} \right) + \\ & \Pi_h \left(\sum_{j=h-k+1}^{h} \frac{q^j}{(1-q)^{j+k-h}} + \frac{V}{W} \sum_{j=h-k}^{h} \frac{q^j}{(1-q)^{j+k-h}} \right) \end{aligned} \tag{5.13}$$

For the cut between $k = 0$ and $k = 1$ the flow equation is

$$\Pi_0 = \frac{V}{W} \left(\sum_{j=1}^{h} \Pi_j q^j \right) \tag{5.14}$$

For the cut between $k = h-1$ and $k = h$ the flow equation changes since the network cannot hold more than h members in the data transmission queue. There is no state to which to increase after Π_h has been reached, in fact

$$\Pi_{h-1} = \Pi_h \left(\frac{V}{W} (\sum_{j=1}^{h} \frac{q^j}{(1-q)^{j-1}} + \sum_{j=2}^{h} \frac{q^j}{(1-q)^{j-1}} \right) \tag{5.15}$$

Eqs. (5.13), (5.14) and (5.15) together with the fact that the sum of all probabilities $\sum_{i=0}^{h} \Pi_i = 1$, form a system with $h+1$ equations with the same number of unknown variables. Starting with Π_{h-1} we can make each of the equations be a function of Π_h. To solve for Π_h, we can substitute each of these terms in the equation $\sum_{i=0}^{h} \Pi_i = 1$. Using Π_h we can get a value for each of the remaining probabilities. Π_0 is the probability that there are no members in the data transmission queue, while Π_h is the probability that the

data transmission queue has reached full capacity. Π_h is important since it represents the fraction of skipped contention periods. Transitions departing from and returning to the same state are omitted in the flow equations, since what comes into the state is the same as what goes out of the state. Thus, the average number of queue members is $\rho = \sum_{i=1}^{h} i\Pi_i$.

The expected duration of the contention period $\overline{B_c}$ is composed of either access periods with no RTS, one RTS, or colliding RTSs. It is also possible to have skipping contention periods. There is no cost for skipping a contention period. Therefore, the duration of an average contention period equals the sum of the percentage P_z of the idle contention periods times their duration $T_i = 2\tau$, plus the percentage P_r of the successfully RTS/CTS exchange periods times their duration $T_s = 2\gamma + 2\tau$, plus the percentage P_c of the tree periods times their duration until the first successful RTS/CTS exchange. This duration is composed of collision steps each with a duration $T_f \leq \gamma + 2\tau$, plus idle steps with a duration of τ seconds and one successful RTS/CTS exchange lasting $T_s = 2\gamma + 2\tau$. The sum of all three portions has to be multiplied by the fraction $1 - \Pi_h$, which represents the fraction of non-skipped contention periods. The tree periods are composed of three parts, each with a distinct cost and duration. Accordingly, the average contention busy period can be written as

$$
\begin{aligned}
\overline{B_c} &\leq \left[T_i P_z + T_s P_r + \left(T_s + (\log(\lambda\tau) + 1)T_f + \frac{T_i}{2}\right) P_c\right](1 - \Pi_h) \\
&= \left[(-\gamma\lambda\tau - \gamma - 2\lambda\tau^2 - 2\tau)\log(\lambda\tau) - 3\tau - \gamma\lambda\tau - 3\lambda\tau^2 - 3\gamma\right](1 - \Pi_h)e^{-\lambda\tau} + \\
&\quad [3\gamma + 5\tau + (\gamma + 2\tau)\log(\lambda\tau)](1 - \Pi_h)
\end{aligned} \tag{5.16}
$$

Substituting Eqs. (5.4), (5.16), (5.5), and (5.6) into Eq. (5.3), we obtain Eq. (5.1) which is the lower bound on the throughput for CARMA-NTQ. ∎

5.3 Numerical Results

The performance comparison is done for both low speed network (9600 bps) and high speed network (1 Mbps) with small data packets of 53 bytes (as in ATM cells) and longer

data packets of 400 bytes. We assume the spacing between stations to be the same and define the diameter of the network to be 16.090 Km., which is 10 miles. Assuming these parameters, the propagation delay of the channel is 54μs. To accommodate the use of IP addresses for destination and source, the minimum size of RTSs and CTSs is 20 bytes. We normalize the throughput of CARMA-NTQ in the same way as we did in Chapter 4 for CARMA.

Network Speed	Packet Size	δ	$a = \frac{\delta}{\tau}$	$b = \frac{\gamma}{\tau}$
9600 bps	424 bits	44166.7 μs	817.9	308.6
9600 bps	3200 bits	333333.3 μs	6172.8	308.6
1 Mbps	424 bits	424 μs	7.85	2.96
1 Mbps	3200 bits	3200 μs	59.3	2.96

Table 5.1: Protocol variables for low-speed networks (9600 bps) and high-speed networks (1 Mbps) with two types of data packets, small (424 bits) or large (3200 bits). The channel delay is $\tau = 54\mu$s, while the control packets are 160 bits long.

If we substitute the new normalized variables from Eq. (4.14) into Eq. (5.1), we obtain

$$S \geq \frac{a\rho}{A'e^{-G} + B'} \tag{5.17}$$

with

$$\begin{aligned} A' &= [(2 + bG + 2G + b)\log(G) + 3 + 3b + 3G + bG](\Pi_h - 1) \\ B' &= (+2 + b)(1 - \Pi_h)\log(G) + (5 + 3b)(1 - \Pi_h) + (2 + \rho) + \frac{\Pi_0}{G} + a\rho \end{aligned} \tag{5.18}$$

In Table 5.1, we summarize the protocol parameters. Fig. 5.3 compares the throughput for low-speed network 9600 bps as well as for high-speed network 1 Mbps for small data packets $\delta = 53$ bytes and large data packets $\delta = 400$ bytes. The throughput is shown versus the offered load (G) for each of the four combinations for CARMA-NTQ. The average

number of queue members ρ is a function of the average number of packets in a message N and can be calculated by solving the probabilities Π_k in the Markov chain.

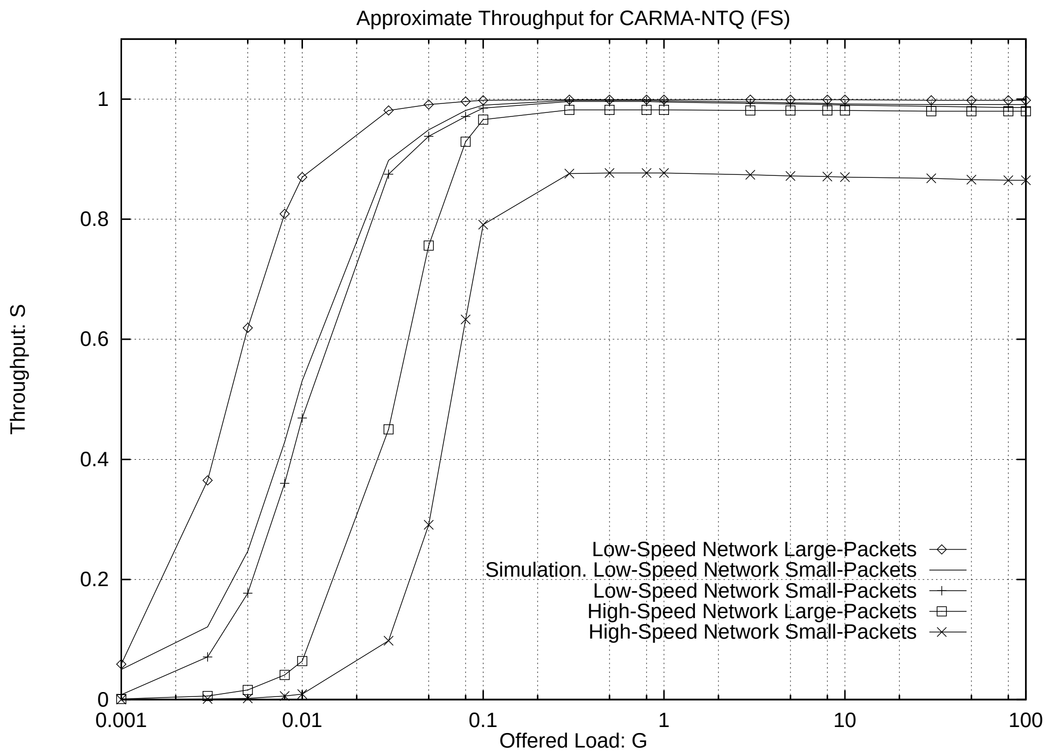

Figure 5.3: Throughput simulation and analysis for CARMA-NTQ (FS).

To verify that the value of S approximated using the upper bounds on average steps for collision resolution times provides a good lower bound for any traffic load, we simulated CARMA-NTQ using 128 stations that generate RTSs according to a Poisson probability distribution function. The maximum number of members in the data transmission queue h was set to 32 and the average number of packets in a message was set to $N = 10$. The simulations were done ten times for each given $m = \tau\lambda$ value to insure convergence. The network speed was 9600 bps with packets of 53 bytes. The simulation results show that our lower bound on the throughput is a good analytical approximation of the throughput. As the average number of members in the data transmission queue increases due to increased offered load, the throughput of CARMA-NTQ approaches the capacity of the channel. The high throughput is achieved by giving each station a share of the channel, each

active station receives the same proportional share. In the event that all stations become active continuously, CARMA-NTQ would allocate the same channel portion as a TDMA framework would for the same number of stations.

5.4 Summary

We have described and analyzed CARMA-NTQ, a new stable multiple access protocol for broadcast channels shared by bursty stations. CARMA-NTQ dynamically divides the channel into cycles of variable length; each cycle consists of a contention period and a transmission-queue period. During the contention period, a station with packets to send competes for the right to be added to the transmission queue; this is done using a collision-resolution-splitting algorithm based on a request-to-send/clear-to-send (RTS/CTS) exchange with carrier sensing.

Our analysis was limited to fully connected networks using a single idle channel, and we have also suggested an approach to apply CARMA-NTQ to WLANs with hidden terminals. Our analysis shows that CARMA-NTQ provides high throughput when either a small or a large number of stations need to access the channel. Allowing the maximum size of the data transmission queue to equal the number of stations in the system, CARMA-NTQ becomes a dynamic polling or token-passing scheme in which the poling sequence or token-passing sequence grows and shrinks with the number of stations with traffic to send. CARMA-NTQ is much more efficient than token-passing and polling schemes when there are only a few stations with packets to send. CARMA-NTQ can operate with no need for time slotting as prior MAC protocols based on collision resolution do.

6. ICRMA

In this chapter we introduce a new stable multiple access protocol for broadcast channels shared by multiple stations, which we call the incremental collision resolution multiple access (ICRMA) protocol. Like previous efficient MAC protocols based on tree-splitting algorithms, ICRMA maintains a distributed queue for the transmission of data packets and a stack for the transmission of control packets used in collision resolution. ICRMA dynamically divides the channel into cycles of variable length; each cycle consists of a contention period and a queue-transmission period. The queue-transmission period is a variable-length train of packets, which are transmitted by stations that have been added to the distributed transmission queue by successfully completing a collision-resolution round in a previous contention period. During the contention period, stations with one or more packets to send compete for the right to be added to the data-transmission queue using a deterministic tree-splitting algorithm. A single round of collision resolution (i.e., a success, and idle or a collision of control packets) is allowed in each contention period. ICRMA is an improvement over CARMA-NTQ in two aspects; access delay and throughput. ICRMA decreases the time for a station to be added to the transmission queue since the collision resolution is done on a continuous basis, one step per contention period. ICRMA does not resolve collisions until one station is added to the queue as CARMA-NTQ does. In ICRMA, a station attempts to join the transmission queue during contention intervals by sending an RTS to any intended receiver, who sends a CTS if the station can join the queue. RTSs are sent according to a deterministic tree-splitting algorithm that resolves all the requests for the queue that arrive during the same contention interval. Access time to the channel is divided into rounds of transmissions for all members of the transmission queue, which we call a queue-transmission period, followed by short contention periods during which stations attempt to join the queue. The queue-transmission period is a variable-length train of packets from stations that have been added to the transmission queue by successfully completing a collision-resolution round in a previous contention period. A single round

of collision resolution (i.e., a success, and idle or a collision of control packets) is allowed in each contention period. The control packets used in each contention period are much smaller than data packets.

Analytical results show that collision resolution in ICRMA is much more efficient than CARMA-NTQ, and simulation and analytical results show that ICRMA's throughput is within 5% of the throughput achieved by the ideal channel access protocol based on a distributed transmission queue and incremental collision resolution.

The chapter is organized as follows. Section 6.1 describes ICRMA. For simplicity, we first describe ICRMA in detail assuming a fully connected network; however, ICRMA can be applied to wireless LANs with hidden terminals using the same extensions described in Chapter 5 for CARMA-NTQ.

Using our bounds derived in Chapter 2 on the number of steps required to resolve a collision of RTSs, Section 6.2 provides a lower bound on the throughput achieved by ICRMA using the model first used by Kleinrock and Tobagi [33] for CSMA. Analytical and simulation results show that the throughput in ICRMA approaches the channel capacity as propagation delays and size of control packets decrease with respect to the size of data packets. The analytical results are very close to the results obtained by simulation, which validates the approximations made in our analysis.

Section 6.3 compares the throughput achieved in ICRMA with that achieved with an ideal channel access protocol based on transmission queues and perfect collision resolution implemented using RTS-CTS exchanges, in which only m successes are needed to resolve m requests for the transmission queue. This analysis shows that, for a given size of control packets and data packets, ICRMA's throughput is within 5% of the optimum throughput.

6.1 ICRMA

This section describes the operation of ICRMA. To simplify our exposition and the analysis of the protocol, we first assume a fully connected network in which all stations

can hear one another. The last part of this section summarizes a simple approach to apply ICRMA to networks with hidden terminals.

6.1.1 Basic Operation

With ICRMA, the channel access time is divided into cycles, with each such cycle consisting of a small, dynamically-sized contention period and a dynamically-sized queue-transmission period. Stations are assumed to monitor the state of the channel while they are not transmitting.

The queue-transmission period consists of the collision-free transmissions from the stations with a position in the transmission queue. Each station allocated to the transmission queue transmits a packet as soon as the packet from the previous station in the queue is received. The maximum spacing between packets is twice the propagation delay. A station wishing to send one or more packets has to acquire a position in the transmission queue before transmitting the data packets.

Stations compete to reserve a position in the transmission queue during the contention periods between queue-transmission periods. Stations follow a non-persistent CSMA strategy for the transmission of RTSs with which they request to be added to the transmission queue. The sender of an RTS to an intended destination listens to the channel for one maximum round-trip time plus the time needed for the destination to send a CTS. If the CTS is not corrupted and is received within the time limit, the station is added to the transmission queue, starting with the next queue-transmission period.

The transmission queue can be allocated a maximum size to provide delay guarantees. Once the transmission queue reaches its maximum length, stations are not allowed to request to be added to the queue, until at least one member of the queue ends transmitting its packets and leaves the queue.

Although each station transmits an RTS only when it determines that the channel is free during a predefined amount of time in the contention period, collisions of RTSs may still occur due to propagation delays. RTSs are vulnerable to collisions for time periods

equal to the propagation delays between the senders of RTSs. ICRMA uses a deterministic tree-splitting algorithm to resolve collisions of RTSs. A single step of this algorithm is taken during each contention period; a step in a contention period can be an idle period, an RTS collision period, or a successful RTS/CTS exchange, all of which are much shorter than a single data packet.

Once the collision-resolution algorithms is started by the collision of RTSs from a set of stations, other stations who want to be added to the transmission queue must wait for all the stations who started the current round of the collision resolution algorithm to be added to the transmission queue through a successful RTS/CTS exchange; they know when this occurs by means of a stack that is maintained distributedly, which is described subsequently. We assume a non-persistent policy in which stations who are not in the transmission queue and are not involved in an collision resolution must backoff for a random period of time if they receive local packets to send and need to be added to the transmission queue.

A station that is not transmitting must be listening to the channel, and knows the current state of the channel. Based on its knowledge of the state of the transmission queue and the current step of the collision-resolution algorithm used to add stations to the queue, a station can be in one of the following states:

- XMIT: the station is part of the transmission queue.
- RTS: the station is trying to acquire a position in the transmission queue and is participating in the collision-resolution algorithm.
- BACKOFF: the station is waiting for a backoff time to complete as a result of receiving local packets to send outside an access period.
- REMOTE: the station is not part of the transmission queue and knows that a set of collisions of requests for the queue are still being resolved.
- PASSIVE: the station is not part of the transmission queue, has no local packets to send, and knows that no prior requests for the queue are being resolved.

6.1.2 Information Maintained and Exchanged

Each station is assigned a unique identifier, knows the maximum number of stations allowed in the network and the maximum propagation delay, and maintains a stack and two variables ($LowID$ and $HiID$). $LowID$ is initially set to 1 and denotes the lowest ID number that is allowed to send an RTS. $HiID$ is the highest ID number that is allowed to send an RTS; it is initially set to the largest number of stations allowed in the network. $LowID$ and $HiID$ constitute the allowed ID-number interval that can send RTSs. If the ID of the station is not within this interval or the station has already been assigned to the data transmission queue, it cannot send its RTS. The stack is simply a storage mechanism for ID-intervals that are waiting for permission to send an RTS.

Each station also maintains the state of the transmission queue, i.e., it knows the members in the transmission queue and their position in the queue and the beginning of each queue-transmission period. RTSs and CTSs specify the IDs of the sender and the intended receiver, as well as the allowed ID-number interval known to the sending station. Including the allowed ID interval in control packets allows listening stations to be updated on the state of the collision resolution. A data packet contains: (a) the user data and destination of the packet; (b) the allowed ID-number interval known to the sending station; (c) the state of the transmission queue, such as the number of stations in the queue and the position of the sending station in the queue; and (d) the state of the transmission for the sending station, such as the length of the current packet, whether the station is leaving the transmission queue, and the length of the *next* packet if variable length packets are allowed. This information updates listening stations quickly on the state of the channel.

6.1.3 Adding Members to the Transmission Queue

Stations can request to be added to the transmission queue only during contention periods. If the maximum length of the transmission-queue is reached, the contention steps are skipped, until space is made available by one or multiple stations leaving the queue. Stations are allowed to send RTSs to be added to the transmission queue only during the

first few seconds of a contention period. This time period is called the *access period* and is used in order to avoid collisions of RTSs with the beginning of the transmission queue. For the purposes of analysis, we assume subsequently that the access period lasts τ seconds, but it is a configurable parameter. A contention period consists of a single step of a deterministic tree-splitting algorithm based on the allowed ID interval, which we describe subsequently.

If a PASSIVE station receives local packets to send during an access period (the first τ seconds of a contention period), it first listens to the channel. If the channel is clear (i.e., no carrier is detected), the station enters the RTS state by sending an RTS. The sender then waits and listens to the channel for one maximum round-trip time plus the time needed for the destination to send a CTS. When the originator receives the CTS from the destination, it is added to the data transmission queue and enters the XMIT state; the station can start sending collision-free packets in the following queue-transmission period.

On the other hand, if a PASSIVE station obtains local packets to send outside an access period, or receives the packets within an access period but detects carrier, it enters the BACKOFF state. The station then computes a random backoff time and attempts to enter the data transmission queue after that time.

If the sender of an RTS does not receive the corresponding CTS within the allocated time, this station and all other stations in the network assume that a collision has occurred. The collision-resolution algorithm is started with the first round of RTS collisions. As soon as a collision is detected, every station divides the ID-interval $(LowID, HiID)$ into two ID-intervals. The first ID-interval, which we will call the backoff interval, is $(LowID, \lceil \frac{HiID+LowID}{2} \rceil - 1)$, while the second ID-interval, the allowed ID interval, is $(\lceil \frac{HiID+LowID}{2} \rceil, HiID)$. Each station in the system updates its stack by executing a PUSH stack command, where the key being pushed is the backoff interval. After this is done, the station updates $LowID$ and $HiID$ with the values from the allowed ID interval and the queue-transmission period follows. This procedure is repeated each time a collision is detected.

Hence, a station knows if collisions are currently being resolved when its local stack is

not empty and the allowed ID interval does not equal the entire ID interval. A PASSIVE station that detects a collision of RTSs goes to the REMOTE state. A station in REMOTE state goes to PASSIVE state as soon as it detects that its local stack is empty and the allowed ID interval is empty. If a station is in REMOTE state and obtains one or more packets to send, it goes immediately to the BACKOFF state.

Only those stations that are in the RTS state participate in the current round of the collision-resolution algorithm to be added to the transmission queue. All stations in the allowed ID-interval that are in the RTS state transmit an RTS in the next contention period. Stations outside of the transmission queue that did not start the current round of collision resolution are forced into REMOTE or BACKOFF state.

Each step of the collision-resolution algorithm, and consequently each contention period, can be one of the following three cases:

- Case 1–*Idle*: There is no station in the RTS state whose ID is within the allowed ID interval. Therefore, the channel must remain idle one collision-resolution step and no new member is assigned to the transmission queue. An idle step of collision resolution lasts a maximum round-trip time; after that time, the stack and the variables $LowID$ and $HiID$ are updated; each station executes a POP command in the stack. This new ID-interval now becomes the new $HiID$ and $LowID$.

- Case 2–*Success*: There is a single station with an RTS to send whose ID lies within the allowed ID interval. In this case, a single station is able to complete an RTS/CTS handshake successfully, enters the XMIT state, and is added to the transmission queue. The duration of a successful collision-resolution step lasts two RTSs and two channel propagation delays.

- Case 3–*Collision*: There are two or more stations with RTSs to send whose IDs are within the allowed ID interval; therefore, each such station sends an RTS creating a collision. Because stations use carrier sensing, a collision period can last no more than an RTS plus a maximum round-trip time. The stations in the allowed ID-interval are

again split into two new ID-intervals; the stack and the variables for each station are updated.

The tree-splitting algorithm executes a collision resolution (idle, success, or collision), until no station remains in the RTS state and all requests for the queue have been resolved. Again, all stations know when the tree-splitting algorithm terminates it current round when the backoff stack is empty and there are no values in the allowed ID interval. Once termination is detected, the allowed ID interval is updated to include all the ID range.

To ensure fairness within the splitting algorithm, the position of the stations in the tree (which is equivalent to changing the ID number) can vary after each tree-contention period. For example, the ID numbers of the stations can rotate cyclically. Each station increases its ID number by one and the last station takes the ID number of the previous first station.

6.1.4 ICRMA Example

The following example illustrates the way in which stations are added to the transmission queue. As we did with CARMA-NTQ, consider the case of a network with four stations and assume that, at time t_0, station n_{11} is the only station assigned to the queue-transmission period with five data packets to transmit and the last cycle has ended and we are at the beginning of a new cycle, that is, in the contention period.

During the access period, packets arrive at stations n_{01} and n_{00}. At time t_0, a collision of RTSs occurs with station n_{00} and n_{01} each sending an RTS in the same contention period, while station n_{10} and station n_{11} do not request to be added to the data transmission queue (see Fig. 6.1). Station n_{11} does not need to send a request, because it is already assigned to the data transmission queue and all other stations know that it still has five data packets to transmit. Let station n_{01} have three data packets to send while station n_{00} has only one data packet to transmit. At time t_0 the first collision of RTSs occurs, all stations in the system notice the beginning of the resolution algorithm, as well as the beginning of the contention period. All stations update their stacks and their $LowID$ as well as their $HiID$ values. After at most an RTS plus a maximum round-trip time has elapsed, the first

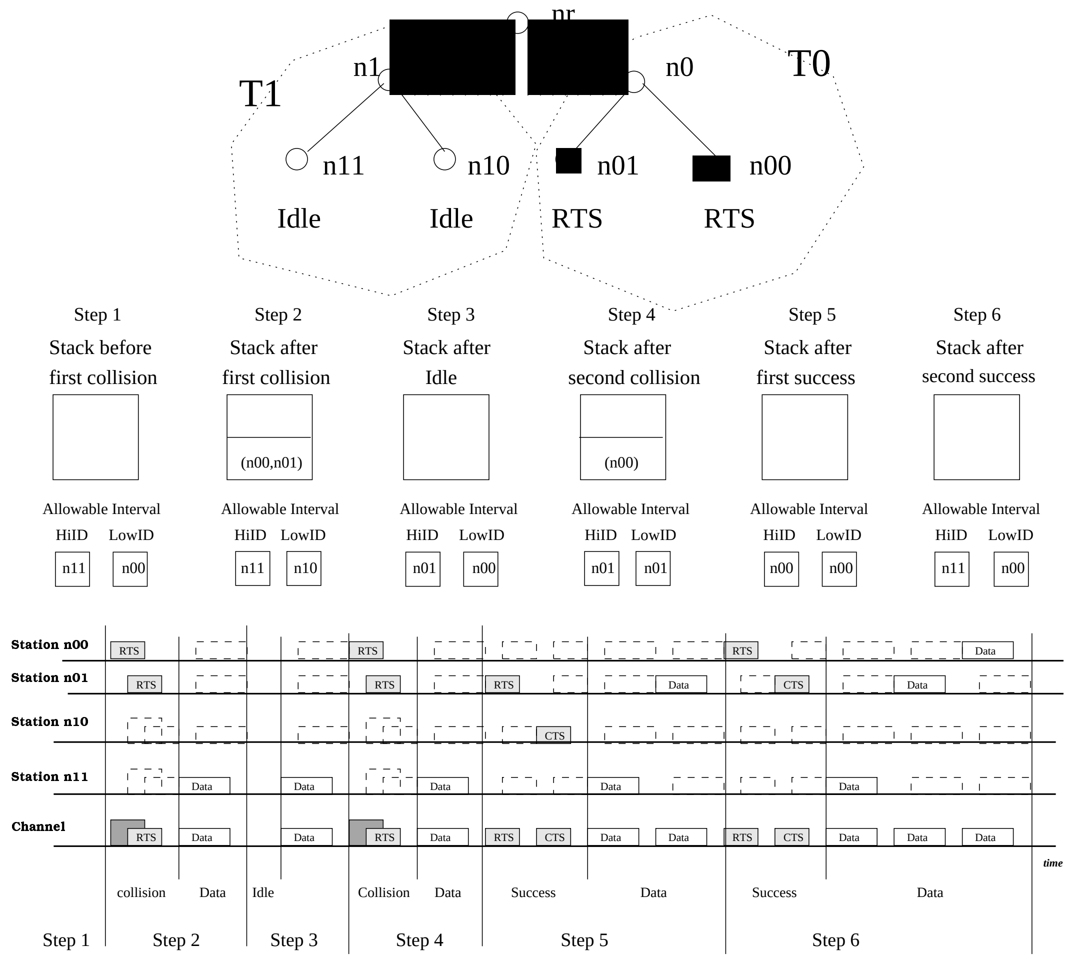

Figure 6.1: Tree structure for an example with $n = 4$ stations and $m = 2$ stations are requesting to be admitted into the data transmission queue. Station n_{11} is already in the queue.

queue-transmission period begins with station n_{11} transmitting a packet. The duration is the size of the length of the packet plus the channel delay. In the next contention period stations n_{00} and n_{01} backoff and wait until the collisions in the allowed-ID interval are resolved. They both are excluded from sending RTSs. Stations n_{10} and n_{11} are allowed to request the channel; an idle contention period occurs because stations n_{11} and n_{10} do not wish to be added to the data transmission queue. After 2τ seconds, all stations notice that the channel is idle, which means that there were no collisions. All the stations in the system must update their intervals and the stack. They execute a POP-stack command and the new

allowed interval is $(00, 01)$; therefore, after n_{11} transmits another packet, the next allowed ID interval can proceed to solve its RTS collisions. Both stations n_{00} and n_{01} transmit an RTS control packet and a collision occurs again. Because a collision occurred, after at most an RTS plus a maximum round-trip time has elapsed the allowed ID interval is split in two. Once again the step in the contention is followed by a queue-transmission period. Station n_{11} is still in the group and has two more packets to send. In the next contention step, station n_{01} is within the allowed interval while the n_{00} station must wait; its interval is the top of the stack. Since the next interval has only one station sending an RTS, station n_{01} is assigned to the data transmission queue after an RTS, a CTS plus a maximum round-trip time. During the successful RTS/CTS exchange, all other stations know that station n_{01} has three packets to send. The contention step ends and station n_{01} begins the transmission of its data packet immediately after it has read the packet transmitted by the preceding group member, which is station n_{11}. After two data packets plus a maximum round-trip time has elapsed, station n_{11} has one data packet left while station n_{01} has two packets. The next contention step is also a successful RTS/CTS exchange; after an RTS, a CTS plus a maximum round-trip time, station n_{00} is added to the group and the allowed interval as well as the stack are empty. The termination of the tree-splitting algorithm is reached when the backoff stack as well as the allowed ID interval is empty. The data transmission queue contains a packet from stations n_{00}, n_{01} and n_{11}. After the last group-transmission period stations n_{00} and n_{11} are done sending packets and free the space within the group. If new packets arrived at one of these stations they will have to request a space in the group. Let us assume that that after termination of the tree-resolution period no new data packets arrive during the access period, therefore, the channel remains idle for two channel delays. During the next queue-transmission period, station n_{01} sends the last packet. Station n_{01} is removed and the data transmission queue is empty. The channel remains idle until a new packet arrives. The allowed ID interval is (n_{00},n_{11}) and the backup stack is empty.

6.1.5 Deleting Members from the Transmission Queue

Deletions from the transmission queue occurs when a station in the XMIT state ends transmitting a packet train an gets out of the queue to remain quiet. In this case, the information contained in a data packet notify the stations when the sending station is sending its last packet before leaving the queue.

Deletions from the transmission queue can also occur due to failures, Failures can be handled by listening to the channel. A maximum interval allowed between two consecutive packets in the queue can be defined based on the maximum propagation delay and processing delays. Because stations are at most τ seconds apart from one another, two consecutive packets in the queue cannot be separated by more than 2τ seconds plus processing delays (including the turn around times of the radios), which we call *maximum queue transmission gap* and assume it lasts ϕ seconds.

A station in the queue must transmit when its turn comes so that its transmission reaches other stations within ϕ seconds of the previous transmission. A "silence turn" could occur in the transmission queue if a station is not ready to transmit a data packet when its turn comes but is not ready to leave the queue. Rather than allowing silence turns to take place, ICRMA requires a station to transmit either a data packet or a short control packet when its turn comes in the queue. The control packet carries no user data and is simply used to convey an update of the state of the queue and the collision resolution process.

6.2 Throughput Analysis

We again assume that there is large number of stations, each forming a Poisson source of RTSs (both new and retransmitted). The aggregate mean generation rate of RTSs by all stations is λ RTSs per unit time. Each station is assumed to have at most one RTS to transmit at any given time. With this model, the average number of RTS arrivals in a time interval τ is $\lambda\tau$, i.e., $m = \lambda\tau$. All data packets have a duration of δ seconds, and the time to transmit a control packet is γ seconds. Both γ and δ are multiples of τ. The number of packets in a message is a random variable, and the probability that a message will complete

its transmission (in a given cycle) is given by $q = \frac{1}{N}$ where N is the average number of packets in a message. We also assume that the time to transition between transmit and receive states is negligible.

The probability P_z that a contention period is idle is equal to the probability that no packets arrive during the access period τ, which can be expressed as

$$P_z = e^{-\lambda\tau} \tag{6.1}$$

For a station to be added to the data transmission queue, the RTS/CTS exchange must be successful, which implies that the RTS must be the only one in the channel during its access period and the data transmission queue must not be full. If there are already h group members, no new member can be added and the contention would be skipped (no station sends an RTS), until a space in the data transmission queue is available. Therefore, the probability P_r that a station sends an RTS successfully equals the probability that only one RTS packet arrives during the τ seconds of the access period, that is,

$$P_r = \lambda\tau e^{-\lambda\tau} \tag{6.2}$$

The probability P_c that a contention period results in an RTS collision is the same as the probability that more than one message arrives during an access period. This value can be expressed as:

$$P_c = 1 - P_r - P_z = 1 - \lambda\tau e^{-\lambda\tau} - e^{-\lambda\tau} \tag{6.3}$$

Making use of the upper bounds from Theorem 6 and Theorem 8, the probability that an RTS/CTS exchange is successful after the tree-splitting algorithm has been started can be approximated by

$$P_a \approx \frac{m}{2.886m - 1} = \frac{\lambda\tau}{2.886\lambda\tau - 1} \approx \frac{1}{3} \tag{6.4}$$

where $m \geq 2$. It is clear that P_a should be a function of both the number of stations that start a round of the tree-splitting algorithm, and the number of collision-resolution steps that have occured up to the successful step. Using an average over all steps in a round of collision-resolution is an approximation we use to simplify our analysis. As we will see this and other approximations we make still render results that are close to those obtained by simulation.

The states of the queue-transmission period can be represented by a Markov chain. The states in the Markov chain represent the probability Π_k that k members are in the data transmission queue. A transition in the chain occurs with each collision resolution step. Notice that the number of members in the data transmission queue can only increase by one, but can decrease by up to k. This is because at most one new member can be added to the data transmission queue per collision-resolution step, but any number of members can be released from the data transmission queue per step. Fig. 6.2 shows an example for a network that allows at most $h = 4$ stations to be members of the group-transmission period. Transitions from one state to the other have been adequately labeled, where

$$W = P_r + P_c P_a \tag{6.5}$$

and

$$V = P_z + P_c(1 - P_a) \tag{6.6}$$

We can generalize our example and define h as the maximum number of members in the data transmission queue allowed by the network. If we draw a line between any two consecutive states of the Markov chain, then the flow of the transitions going in one direction has to be equal to the flow in the other direction. Therefore, for any cut within the Markov chain excluding the first and the last state, i.e., for $1 \leq k \leq h-2$,

$$\Pi_k W(1-q)^k \quad = \quad \sum_{i=1}^{h-k-1} \Pi_{k+i} \left(W \sum_{j=i+1}^{k+i} q^j (1-q)^{k+i-j} + V \sum_{j=i}^{k+i} q^j (1-q)^{k+i-j} \right) +$$

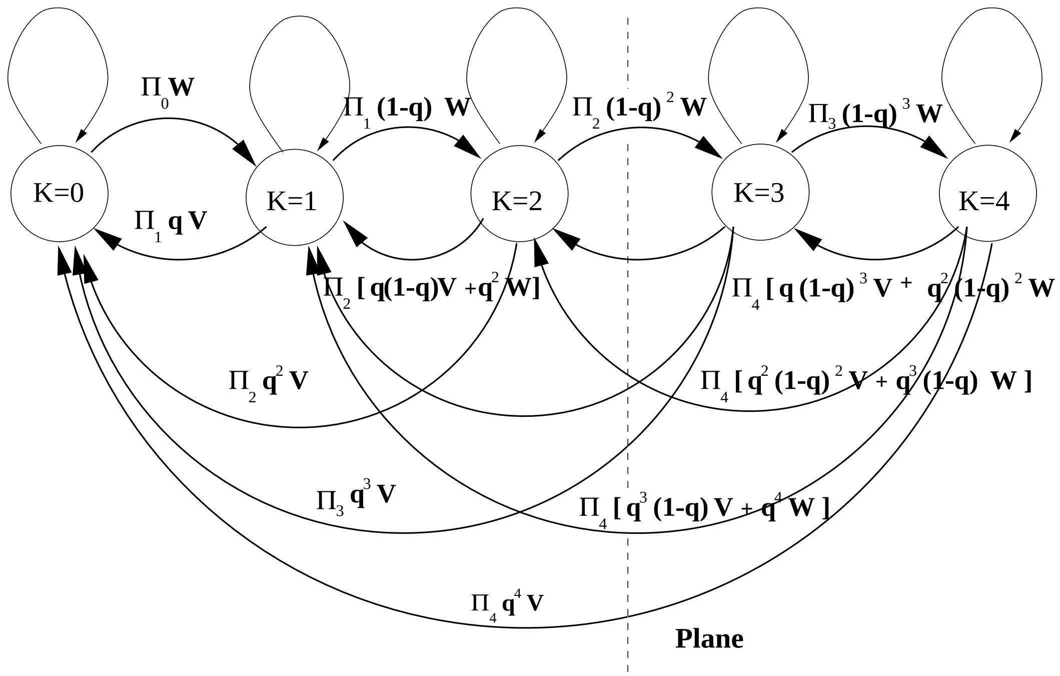

Figure 6.2: Markov Chain defining the transitions from one state to the others. The given example is for a network that allows, up to four members in the data transmission queue. Only a subset of the transition probabilities are shown

$$\Pi_h \left(W \sum_{j=h-k+1}^{h} q^j (1-q)^{h-j} + V \sum_{j=h-k}^{h} q^j (1-q)^{h-j} \right) \tag{6.7}$$

If we divide both sides by $W(1-q)^k$, the result is

$$\begin{aligned} \Pi_k \quad = \quad & \sum_{i=1}^{h-k-1} \Pi_{k+i} \left(\sum_{j=i+1}^{k+i} \frac{q^j}{(1-q)^{j-i}} + \frac{V}{W} \sum_{j=i}^{k+i} \frac{q^j}{(1-q)^{j-i}} \right) + \\ & \Pi_h \left(\sum_{j=h-k+1}^{h} \frac{q^j}{(1-q)^{j+k-h}} + \frac{V}{W} \sum_{j=h-k}^{h} \frac{q^j}{(1-q)^{j+k-h}} \right) \end{aligned} \tag{6.8}$$

For the cut between $k = 0$ and $k = 1$ the flow equation is

$$\Pi_0 = \frac{V}{W} \left(\sum_{j=1}^{h} \Pi_j q^j \right) \tag{6.9}$$

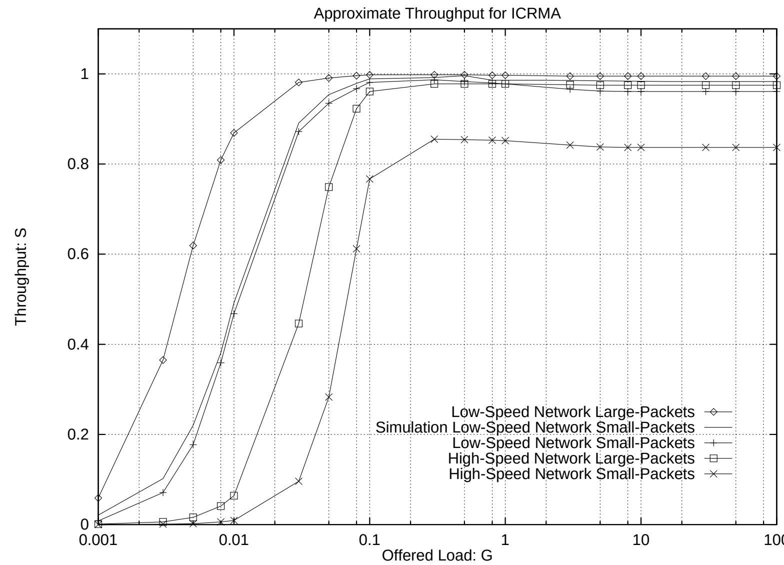

Figure 6.3: Approximate throughput comparison for ICRMA for low-speed and high-speed networks with small and large data packets. The maximum number of group members is $h = 32$ and the average number of packets in a message is $N = 10$.

For the cut between $k = h - 1$ and $k = h$ the flow equation changes since the network cannot hold more than h members in the data transmission queue. There is no state to which to increase after Π_h has been reached, in fact

$$\Pi_{h-1} = \Pi_h \left(\frac{V}{W} (\sum_{j=1}^{h} \frac{q^j}{(1-q)^{j-1}} + \sum_{j=2}^{h} \frac{q^j}{(1-q)^{j-1}} \right) \tag{6.10}$$

Eqs. (6.8), (6.9) and (6.10) together with the fact that the sum of all probabilities $\sum_{i=0}^{h} \Pi_i = 1$, form a system with $h + 1$ equations with the same number of unknown variables. Starting with Π_{h-1} we can make each of the equations be a function of Π_h. To solve for Π_h, we can substitute each of these terms in the equation $\sum_{i=0}^{h} \Pi_i = 1$. Using Π_h we can get a value for each of the remaining probabilities. Π_0 is the probability that there are no members in the data transmission queue, while Π_h is the probability that the data transmission queue has reached full capacity. Π_h is important since it represents the fraction of skipped contention periods. Transitions departing from and returning to the

same state are omitted in the flow equations, since what comes into the state is the same as what goes out of the state. Thus, the average number of group members is $\rho = \sum_{i=1}^{h} i\Pi_i$.

The average channel throughput is equal to

$$S = \frac{\overline{U}}{\overline{B_c} + \overline{B_g} + \overline{I}} \tag{6.11}$$

where $\overline{U}$ is the average utilization time of the channel, during which the channel is being used to transmit data packets; $\overline{B_c}$ is the expected duration of the contention period; $\overline{B_g}$ is the expected duration of the queue-transmission period; and $\overline{I}$ is the average idle period, i.e., the average interval between two consecutive busy periods.

Because data packets are send free of collisions, $\overline{U}$ equals on the average number of members in the data transmission queue (ρ) times δ, which is the time spent sending a data packet, that is,

$$\overline{U} = \rho\delta \tag{6.12}$$

A contention period is skipped if the data transmission queue is full; otherwise, if there is room in the queue, which is the case with probability $1 - \Pi_h$, a contention period can be idle, successful, or have RTS collisions. The duration of an idle contention period is $T_i = 2\tau$, the duration of a successfully RTS/CTS exchange period is $T_s = 2\gamma + 2\tau$, and the duration collision period is $T_f \leq \gamma + 2\tau$. Therefore, the duration of the contention period is at most $2\gamma + 2\tau$ when it is not idle. Accordingly, the average duration of a contention busy period can be bounded by

$$\overline{B_c} \quad \leq \quad [2\tau P_z + T_s(P_r + P_c)]\,(1 - \Pi_h) \tag{6.13}$$

The average queue-transmission period is equal to the average number of group members (ρ) times the transmission period for one packet ($\delta + \tau$); therefore,

$$\overline{B_g} \;=\; \rho \cdot (\delta + \tau) \tag{6.14}$$

If the data transmission queue is empty, the length of the idle period is given by the next RTS arrival into the channel plus a waiting period of 2τ. In contrast, when the data transmission queue is not empty, the length of the idle period is limited by the start of the next transmission period, which is 2τ seconds. Accordingly, we obtain

$$\overline{I} \;=\; \Pi_0 \left(\frac{1}{\lambda} + 2\tau\right) + (1 - \Pi_0)2\tau = \frac{1}{\lambda}\Pi_0 + 2\tau \tag{6.15}$$

Substituting Eqs. (6.12), (6.13), (6.14), and (6.15) into Eq. (6.11) the throughput can be written as:

$$S \geq \frac{\rho\delta}{-2\gamma(1-\Pi_h)e^{-\lambda\tau} + 2\gamma(1-\Pi_h) + \tau(4 - 2\Pi_h) + \rho(\delta+\tau) + \frac{\Pi_0}{\lambda}} \tag{6.16}$$

The performance comparison is done for both a low-speed network (9600 bps) and a high-peed network (1 Mbps) with small data packets of 53 bytes (as in ATM cells) and longer data packets of 400 bytes. We assume the spacing between stations to be the same and define the diameter of the network to be 16.090 Km., which is 10 miles. Assuming these parameters, the propagation delay of the channel is 54μs. To accommodate the use of IP addresses for destination and source, the minimum size of RTSs and CTSs is 20 bytes. We normalize the throughput of ICRMA in the same way we did for the case of CARMA in Chapter 4. If we substitute the new normalized variables from Eq. (4.14) into Eq. (6.16), we obtain

$$S \geq \frac{a\rho}{-2b(1-\Pi_h)e^{-G} + 2b(1-\Pi_h) + 4 - 2\Pi_h + (a+1)\rho + \frac{\Pi_0}{G}} \tag{6.17}$$

Fig. 6.3 shows the throughput versus offered load give by Eq. (6.17). We consider a low-speed network at 9600 bps, as well as a high-speed network at 1 Mbps using small

data packets ($\delta = 53$ bytes) and large data packets ($\delta = 400$ bytes). The throughput is for each of the four combinations. As the average number of members in the data transmission queue increases due to increases in the offered load, the throughput of ICRMA reaches a constant throughput value that, approaches the channel capacity as the relative overhead due to propagation delays and control packets approaches 0. By taking the limit of S as λ tends to infinity, we obtain

$$\lim_{\lambda \to \infty} S \geq \frac{h\delta}{2\tau + h(\delta + \tau)}$$

which clearly shows that ICRMA's overhead is very small at high load.

To verify that the value of S approximated using the upper bounds on average steps for collision resolution times provides a good lower bound for any traffic load, we simulated ICRMA using 128 stations that generate RTSs according to a Poisson probability distribution function. The maximum number of members allowed in the data transmission queue h was set to 32 and the average number of packets in a message was set to $N = 10$. The simulations were done ten times for each given $m = \tau\lambda$ value to insure convergence. The results of the simulation are shown in Figure 6.3 and indicate that our analysis provides a very good lower of the throughput.

6.3 Comparison with an Optimum Queue Protocol

An optimum MAC protocol based on a distributed transmission queue and RTS/CTS handshakes for additions to the transmission queue would require exactly m successful RTS/CTS handshakes to resolve m requests for additions to the transmission queue. In this section, we obtain the throughput for such an optimum MAC protocol using the model and results derived in the previous section. We then show that ICRMA is indeed very close to the best possible performance.

In terms of our analysis in the previous section, the probability P_a that an RTS/CTS exchange is successful in the contention period when space is available in the transmission

queue is always 1. Therefore, the parameters W and V for the Markov chain (see Fig. 6.2) become

$$W = P_r + P_c \tag{6.18}$$

and

$$V = P_z \tag{6.19}$$

All other parameters in the Markov chain remain the same. If there are multiple stations with RTSs at the beginning of the round of collision resolution, it is assumed that the protocol organizes the transmission of RTSs in a way that success is ensured at each step. Accordingly, the collision-resolution steps for the optimum MAC protocol consist of successful RTS/CTS exchanges only and the average contention busy period in Eq. (6.13) can be rewritten as:

$$\overline{B_c} = [2\tau P_z + T_s(P_r + P_c)]\,(1 - \Pi_h) \tag{6.20}$$

The rest of the variables in Eq. (6.11) remain unchanged. Substituting Eqs. (6.12), (6.20), (6.14), and (6.15) into Eq. (6.11) the average normalized throughput for the perfect protocol can be written as:

$$S_p = \frac{a\rho}{-2b(1-\Pi_h)e^{-G} + 2b(1-\Pi_h) + 4 - 2\Pi_h + (a+1)\rho + \frac{\Pi_0}{G}} \tag{6.21}$$

Notice that the throughput equations are the same as ICRMA's with the exception that average number of members in the transmission queue (ρ) as well as Π_h and Π_0 are different and are determined by the Markov chain. The average number of members in the transmission queue (ρ) increases much faster in case of the optimum queue protocol.

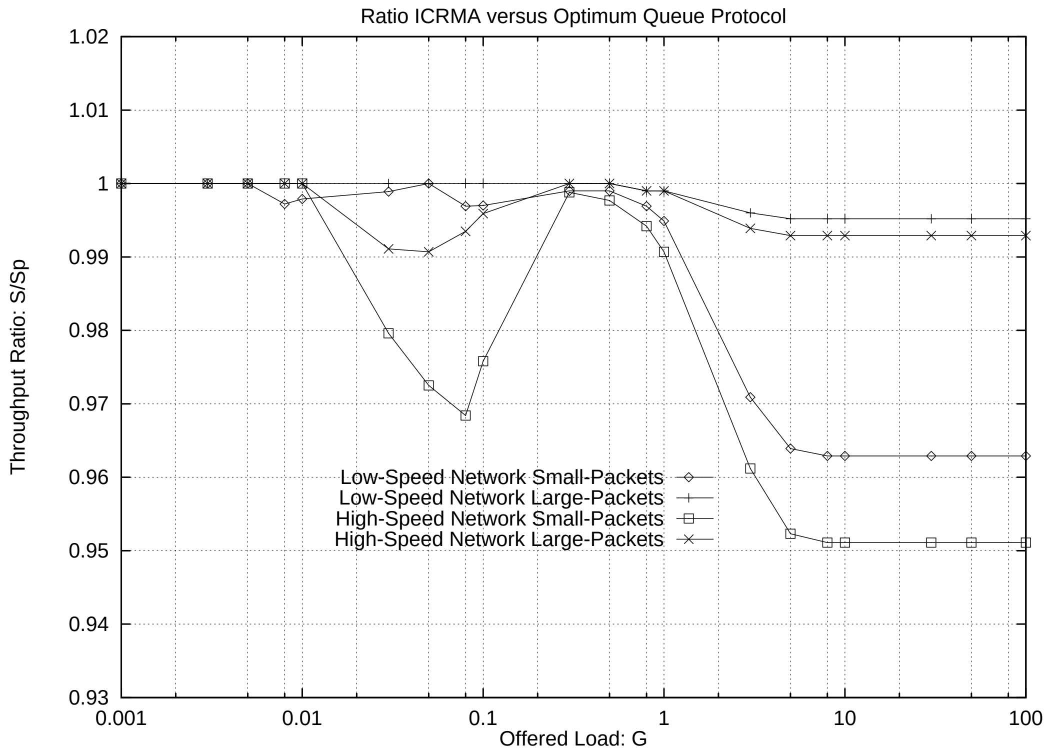

Figure 6.4: Throughput achieved in ICRMA with that achieved with an ideal channel access protocol based on transmission queues and a perfect collision resolution as function of the offer load G. The maximum number of group members is $h = 32$ and the average number of packets in a message is $N = 10$.

Eq. (6.21) is an exact value of the throughput, not an approximation as for the case of ICRMA, because we have used exact values for P_a and the duration of the busy period. Fig. 6.4 shows the ratio of the lower bound of ICRMA's throughput given by Eq. (6.17) to the throughput obtained by the optimum queue protocol given by Eq. (6.21) as the offered load G varies. It is evident that ICRMA approaches optimum average performance. ICRMA differs the most from the optimum MAC protocol when the average transmission queue size is small and requests for addition to the queue (RTSs) accumulate in contention periods. For the case of a high-speed network with small packets, this occurs when G is in the neighborhood of 0.1, and even in this case ICRMA exhibits an throughput within 5% of the optimum. When data packets are much larger than RTS and CTS, ICRMA is within

1% of the optimum.

6.4 Summary

We have described and analyzed ICRMA, a new stable multiple access protocol for broadcast channels shared by bursty stations. ICRMA dynamically divides the channel into cycles of variable length; each cycle consists of a contention period and a transmission-queue period. During the contention period, a station with one or more packets to send competes for the right to be added to the transmission group; this is done using a collision-resolution-splitting algorithm based on a request-to-send/clear-to-send (RTS/CTS) exchange with carrier sensing.

Our analysis was limited to fully connected networks using a single channel, and shows that ICRMA provides high throughput when either a small or a large number of stations need to access the channel. Allowing the maximum size of the data transmission queue to equal the number of stations in the system, ICRMA becomes TDMA in effect when all stations need to transmit. ICRMA is much more efficient than DQRAP, which is a representative of prior efficient protocols based on collision resolution. Furthermore, we have shown that ICRMA exhibits near-optimum performance. In addition to its efficiency, ICRMA is attractive, because it can operate with no need for time slotting as prior MAC protocols based on collision resolution do. The latency in ICRMA is shorter then for CARMA-NTQ, because the contention period is at most one collision-resolution step. A station in the transmission queue does not have to wait long before transmitting the next packet. The latency in ICRMA is not worst than the latency for TDMA.

7. CARMA-MC

This chapter presents multichannel extensions of the basic collision avoidance reservation scheme proposed in Chapter 4. The proposed protocol eliminates busy interference and co-channel interference, and mitigates the effects of contention interference. The objective of using multichannel is to permit collision resolution to operate in ad-hoc networks.

Multichannel networks are more complex than single-channel networks. Multichannel networks can be derived using unique orthogonal codes, as in direct sequence spread spectrum (DSSS) and in code division multiple access (CDMA). Unique orthogonal codes can also be established from a larger frequency domain which is split into several smaller non-overlapping sub-channels as in frequency hopping spread spectrum (FHSS) or as in frequency domain multiple access (FDMA). Orthogonal channels in packet-radio networks permit multiple stations within range of the same receivers to transmit concurrently without interference.

Multiple channels exhibit better delay characteristics than single-channel networks [35, 37, 39] and have better fault tolerance against fading and noise [11, 37]. One drawback of multiple channel networks is that not all the channels may be used efficiently.

In the past, multichannel networks have been constructed using multiple transceivers operating on separate fixed channels [49]. Current transceiver technology has made radio devices available with as many as 300 channels. This allows multichannel networks to be constructed inexpensively using a single device at each station.

Early work in protocol design for multichannel networks used CSMA or ALOHA protocols in slotted multiple channels [38]. A reservation protocol over multiple channels is investigated in [35] for satellite communication systems. A sequential multichannel system which uses CSMA/CA on each channel to dynamically assign stations to channels is presented in [8]. Analysis of multi-hop multichannel networks using CDMA in sparse networks with receiver-based, transmitter-based, pairwise-based, and common channel assignment is presented in [28].

In this chapter we introduce a new stable receiver-initiated, multi-hop, multichannel, multiple access protocol with collision resolution called **C**ollision **A**voidance and **R**esolution **M**ultiple **A**ccess **M**ulti**C**hannel (CARMA-MC) protocol. CARMA-MC is a receiver initiated protocol that dynamically divides the channel into cycles of variable length where each cycle consists of a receiving period and a transmission period. The transmission period has a variable-length duration and each receiving period consists of collision resolution steps, i.e., of success, idle or collision of control packets. The protocol maintains a stack for the transmission of control packets used in collision resolution.

During the receiving period CARMA-MC uses a deterministic tree-splitting algorithm and an RTR/RTS/CTS exchange. A receiving period is initiated by the receiver sending a ready-to-receive (RTR) signal and takes place in the channel assigned to the receiver station. During contention intervals a station attempts to acquire the floor by sending an RTS to the intended receiver who, in turn, sends a CTS if the received RTS is error-free. RTSs are sent according to a deterministic tree-splitting algorithm that resolves all the requests that arrive during the same receiving interval. A variable-length train of packets is transmitted from the station that has acquired the floor by successfully completing a collision-resolution round. The control packets used in each contention step are much smaller than a single data packet.

The chapter is organized as follows. In Section 7.1 we provide some definitions and assumptions for CARMA-MC. Section 7.2 describes CARMA-MC in detail. In Section 7.3 we present the worst case packet delay and channel utilization. Section 7.4 compares the analytical results with the simulation. The analytical results are very close to the results obtained by the simulation, and this validates the approximations made in the analysis.

7.1 Definitions and Assumptions

We define d_{max} as the maximum number of one-hop neighbors that each station has. We assume that there exists a mechanism that ensures that no station is assigned the same receiving frequency as any of its two-hop neighbors. The assignment of codes or channels

can be achieved by applying the distributed assignment of codes algorithm proposed in [25]. Let n be the number of distinct codes such that no code is repeated within by any station two-hops away. Observe that n is also the maximum number of distinct leaves in the collision-resolution tree. This mechanism eliminates co-channel interference.

If a station is active it can be either in the RECEIVER-CONTENTION state or in the SENDER-CONTENTION state. A station in the RECEIVER-CONTENTION state initiates the collision-resolution interval (CRI) in its assigned channel and at the same time it is the intended receiver for its one-hop neighbors. CRI are composed of idle, collision, and successful steps. The beginning of a CRI is initiated by an RTR and an allowed ID interval that includes all the one-hop neighbors. The duration of an RTR packet is ρ seconds. A CRI interval is terminated once the backoff stack and the allowed ID interval are empty. A station in the SENDER-CONTENTION state participates on the CRI initiated by its intended receiver and tries to acquire the floor in order to transmit its data packet.

7.2 CARMA-MC

7.2.1 Basic Operation

Each station uses its assigned channel to initiate data transmissions and at the same time it is the receiver of its one-hop neighbors. Besides the assignment of a unique channel, each station is also assigned a unique identifier (ID), a stack, and two variables $LowID$ and $HiID$ as described in Chapter 4 for CARMA. The allowed ID interval is broadcasted by the receiver to all its one hop neighbors in a ready-to-receive packet (RTR). An RTR is transmitted by the receiver at the beginning of every contention step. The allowed ID interval as well the unique receiver ID are embedded within the RTR.

In CARMA-MS an active station can be either a receiver or a sender. A station is in the receiving mode unless it wants to transmit a data packet on another channel. Once a station is a receiver it remains in that state until the end of the current CRI. Now, two cases must be considered depending on whether or not the station has a data packet to send. If at the end of the current CRI the station has a data packet train to send then it switches

to the transmission mode, i.e., it becomes a sender, and competes to acquire the floor on the channel of its intended receiver. Otherwise, if at the end of the CRI the station does not have a packet train to send, then the station will remain a receiver and send an RTR message at the beginning of the next CRI. Any station engaged in a CRI as a sender must remain in the receiver's channel for at least the duration of a maximum CRI.

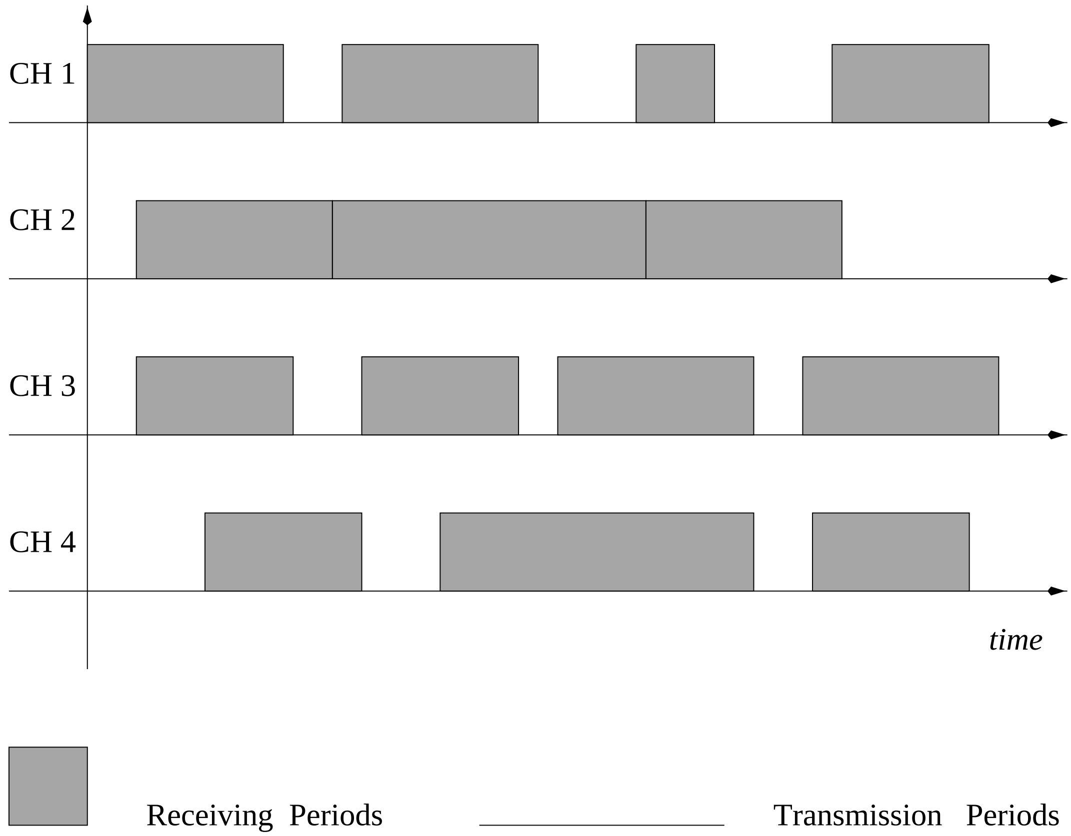

Figure 7.1: Each channel is composed of receiving periods and transmission periods.

As shown in Fig. 7.1, the channel access time in CARMA-MC is divided into cycles, consisting of receiving periods and transmission periods. The receiving period is a sequence of collision-resolution steps each initiated by an RTR. Stations wishing to acquire the floor are assumed to constantly monitor the state of the channel while they are not transmitting. It is assumed that all one hop neighbors are at most τ seconds apart from each other. The duration of a contention step varies according to the type of the collision-resolution step.

It is possible for a channel to have only receiving periods. This is the case if the station owner of the channel does not have data packets to send and is active. This station will stay in its RECEIVER-CONTENTION state constantly initiating new CRI as is the case for in channel 2 in Fig. 7.1.

If a station has no local packet pending, then the station can initiate the receiving period by transmitting an RTR on the channel assigned to it. The station becomes a receiver station for its one-hop neighbors by transmitting an RTR at the beginning of each contention-resolution step, i.e., the station makes a transition to the RECEIVER-CONTENTION state. Each RTR can be visualize as a small packet indicating to other stations that the station transmitting the RTR is ready to receive a request for the floor. The RTR also contains information regarding the allowed ID interval of stations that compete to acquire the floor. Since only the receiver station sends an RTR in its corresponding channel, collisions of RTR can never occur. The RECEIVER-CONTENTION state is the default state for all active stations.

A station in the RECEIVER-CONTENTION state with a local packet pending makes a transition to the SENDER-CONTENTION state, if the current CRI is completed, i.e., the station must resolve all the contentions for the current CRI in its channel. Only then, it scans the destination channel for at most the maximum duration of a CRI. If an RTR is heard during this interval and the station's ID is within the allowed ID interval, then the station competes to acquire the floor extending its duration in the intended receivers channel until it acquires the floor.

A station acquires the floor by sending an RTS. The station follows a non-persistent CSMA strategy for the transmission of RTSs. More precisely, the RTS is directed to the receiver in the channel where the RTR was sensed. The sender of an RTS waits and listens for one maximum round-trip time, plus the time needed for the destination's CTS to arrive. If the CTS is not corrupted and is received within the time limit, then the station acquires the floor and transmits its data packet or train of data packets. If the CTS is not received, all stations monitoring the channel detect a collision. Notice that a station wishing to acquire

the floor spends at most the equivalent of two maximum CRI durations. The maximum CRI duration is determined by the maximum number of one-hop neighbors allowed. The rule is that if within the first maximum CRI duration the station does not hear a valid RTR, the station transitions back to the RECEIVER-CONTENTION state and remains in this state until the end of a CRI.

Although each station transmits an RTS only after the RTR is received and only if the station is within the allowed ID interval, collisions of RTSs may still occur due to propagation delays. RTSs are vulnerable to collisions for time periods equal to the propagation delays between the senders of RTSs. CARMA-MC uses a deterministic tree-splitting algorithm [18] to resolve collisions of RTSs, which are detected by the absence of a CTS. Each step of the CRI is initiated by an RTR. There are three types of steps, idle, collision, and success. An idle steps has a duration of $\rho + 2\tau$, a collision step has a duration of $\rho + \gamma + 3\tau$, and a success step has a duration of $\rho + 2\gamma + \delta + 4\tau$, where ρ is the size of an RTR, γ is the size of an RTS or a CTS, δ is the size of a data packet, and τ is the maximum channel delay.

7.2.2 Information Maintained and Exchanged

Information is maintained, exchanged, and broadcasted by the receiver station. Each station is assigned a unique identifier, a unique channel within a radius of two hops, and maintains a stack and two variables $LowID$ and $HiID$. Recall that $LowID$ is the lowest ID number that is allowed to send an RTS and it is initially set to 1. On the other hand, $HiID$ is the highest ID number that is allowed to send an RTS and it is initially set to the largest number of one-hop stations allowed in the network. $LowID$ and $HiID$ define the allowed ID-number interval, $(LowID, HiID)$, that can send RTSs. If the ID of a station is not within the allowed ID interval, then the station is not allowed to send its RTS. The RTSs and CTSs specify the IDs of the sender and of the intended receiver. Finally, the stack is the storage mechanism for ID intervals that are waiting for permission to send an RTS.

Throughout the chapter we assume that each station knows the maximum number of one-hop stations allowed in the network and the maximum propagation delay.

7.2.3 Acquiring the Floor

As mentioned in Section 7.2.1, stations can acquire the floor by sending an RTS. Since collisions of RTSs may occur due to propagation delays, CARMA-MC uses a deterministic tree-splitting algorithm to resolve such collisions.

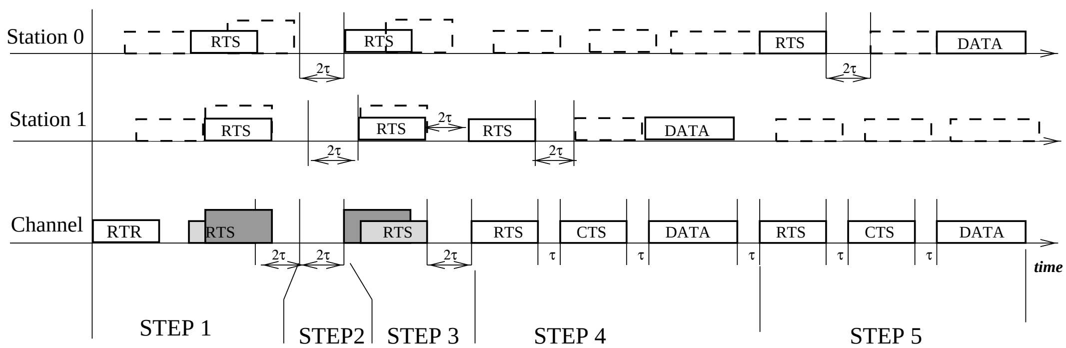

Figure 7.2: Receiving mode: The intended receiver sends an RTR to initiate the CRI. Notice that we have omitted the RTR packet at the beginning of each collision-resolution step.

Whenever a station has local packets to send the following operations are executed. First, the station scans the intended receivers channel for an RTR. If the RTR is detected, the station sends an RTS and it waits and listens to the channel for one maximum round-trip time plus the time needed for the destination to send a CTS. Now, two cases must be considered depending on whether or not the CTS is received by the sender. If the CTS is received within the allocated time, then the sender acquires the floor and can start sending collision-free packets. On the other hand, if the CTS is not received within the allocated time, then the station sender of the RTS and all the other stations competing for the floor detect a collision. In this case, the collision-resolution algorithm is started with the first round of RTS collisions. As soon as a collision is detected, the receiver station divides the allowed ID interval $(LowID, HiID)$ into two ID intervals. The first ID interval, called

the backoff interval, is $(LowID, \lceil\frac{HiID+LowID}{2}\rceil - 1)$, while the second ID interval, which is the new allowed ID interval, is $(\lceil\frac{HiID+LowID}{2}\rceil, HiID)$. The receiver updates its stack by executing a PUSH stack command, where the key being pushed is the backoff interval. After this is done, the station updates $LowID$ and $HiID$ with the values from the new allowed ID interval and the queue-transmission period follows. This procedure is repeated each time a collision is detected. This information is broadcasted by the receiver in the RTR in the next contention step. Each station in the current channel receives this information, and thus becomes aware of the current state.

Recall that each step of the collision-resolution algorithm, and consequently each contention step, can be either:

- Case 1–*Idle*: There is no station in the RTS state whose ID is within the allowed ID interval. The channel remains idle for the duration of one propagation delay. The stack and the variables $LowID$ and $HiID$ are updated. The receiver station executes a POP command in the stack updating the allowed ID interval with the new values.
- Case 2–*Success*: There is a single station with an RTS to send whose ID lies within the allowed ID interval. In this case, a single station is able to complete an RTS/CTS handshake successfully, acquiring the floor and transmitting its data packet.
- Case 3–*Collision*: There are two or more stations with RTSs to send whose IDs are within the allowed ID interval. In this case, each of these stations sends an RTS creating a collision. The stations in the allowed ID interval are again split into two new ID intervals and the stack and the variables for the receiver station are updated.

7.3 Channel Utilization and Packet Delay

In this section we derive a lower bound on the average utilization of the channel as well as an upper bound on the average delay that a station experiences in transmitting a data packet. Recall that, the channel utilization is defined as the amount of time that the channel is used to transmit data packets divided by the total time.

In order to simplify our analysis, we assume that each station has at most d_{max} one-hop neighbors and that each station has a different receiving frequency or channel than its two-hop neighbors. Furthermore, we assume that all the channels have similar behaviors and this enable us to study the utilization of a generic channel. Observe that, if $d_{max} > 2$ then the number of distinct receiving channels is $n = d_{max}^2 - d_{max} - 2$.

Let ch_r denote the receiving channel, f_r the frequency associated with channel ch_r and n_r the receiving station. Then, station n_r is the intended receiver for at most d_{max} one-hop neighbors. At the same time, any of the d_{max} one-hop neighbors is a potential receiver for station n_r. We also let n_x denote the intended receiver of n_r regardless of which of the d_{max} one-hop neighbors station n_r chooses to send its data packet. Thus, f_x is the transmitting frequency of station n_r and ch_x is the channel associated with frequency f_x. Finally, let $T(n, d_{max})$ be the maximum possible duration of a CRI.

Recall that a channel is composed of receiving intervals followed by transmitting intervals. Once a station is in the receiving mode (i.e., is a receiver) it remains in that mode until the end of the current CRI. Only then, the station can switch to the transmitting mode if its transmitting queue contains data packets that need to be sent. We further assume that when a station acquires the floor it can only transmit one data packet.

We start by computing in Theorem 15 an upper bound on the average delay that a station experiences in transmitting a packet. According to the CARMA-MC protocol, a station trying to transmit a packet can wait at most $T(n, d_{max})$ time for an RTR. If during this maximum period of time it does not hear an RTR, it must return to its channel and initiate a CRI. On the other hand, if it hears an RTR it will transmit its packet within an additional $T(n, d_{max})$ interval. This implies that the duration of the longest possible transmission period is $2T(n, d_{max})$. Thus, for our protocol the best possible receiver is the one that spends all its time in the receiving mode, i.e., its transmission queue is always empty, and a station requesting the floor in the channel of such receiver will always succeed in the request within the time interval $T(n, d_{max})$. On the other hand, the worst possible receiver is the one that spends most of its time in the transmitting mode and the least

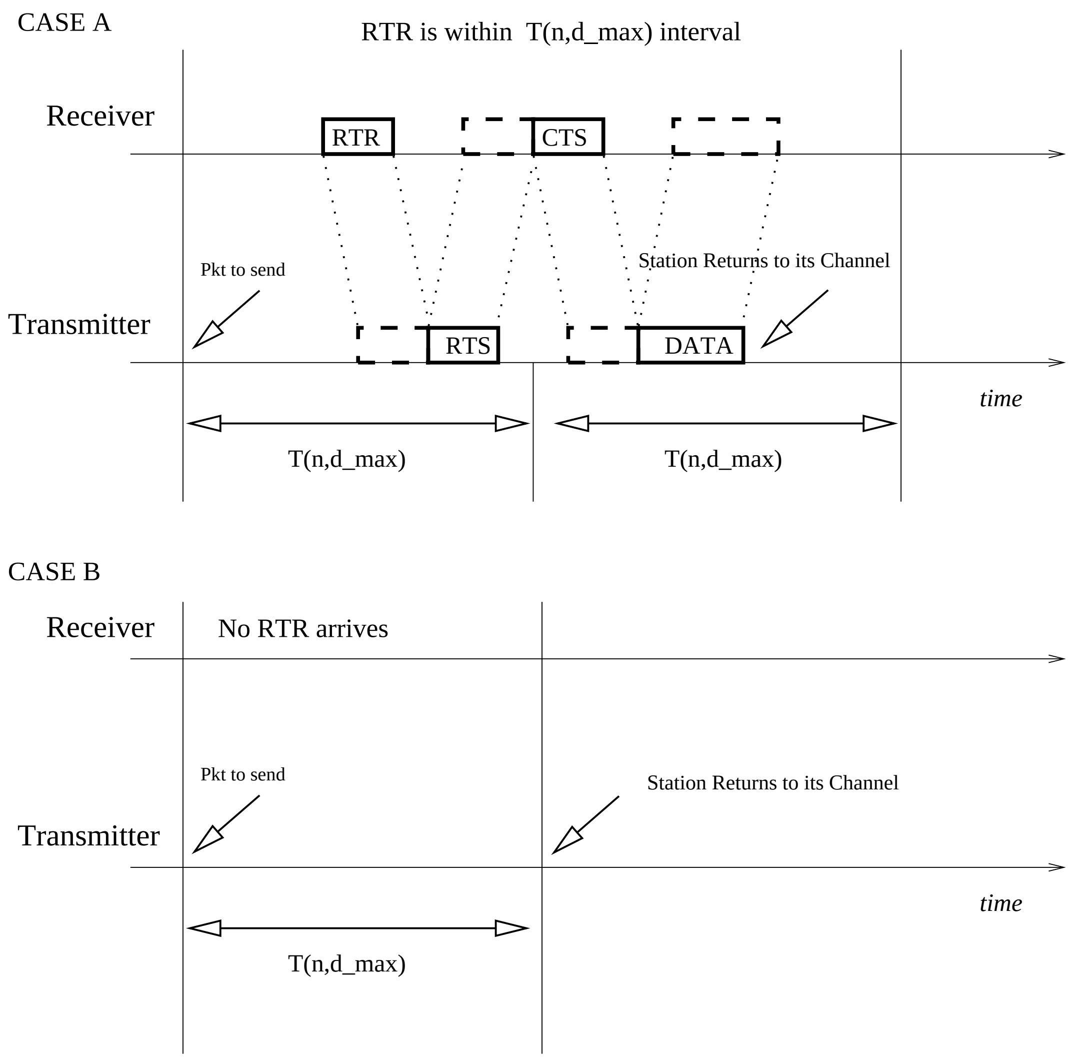

Figure 7.3: Transmission period for CARMA-MC. In case A, the RTR arrives within the the time interval $T(n, d_{max})$, therefore, the sender contents for the floor. In case B, the RTR does not arrived within the allowable interval, therefore, the sender must return to its channel and transition to the receiving mode.

amount of time in the receiving mode. That is, the worst possible receiver will constantly have idle steps (i.e., non of its neighbors will request the floor) and receive a data packet in its transmission queue when found in the RECEIVER-CONTENTION state. This will force the receiver to switch to the SENDER-CONTENTION state and then to remain in this state for as long as possible. Now, since the duration of an idle step is $\rho + 2\tau$ where

ρ is the size of the RTR packet and τ is the maximum channels delay, and the duration of the longest possible transmission period is $2T(n, d_{max})$, each time a sender targets the worst possible receiver its probability of success is

$$P_s = \frac{T(n, d_{max})}{2T(n, d_{max}) + \rho + 2\tau} < 0.5 \tag{7.1}$$

and its probability of failure is

$$P_f = \frac{T(n, d_{max}) + \rho + 2\tau}{2T(n, d_{max}) + \rho + 2\tau} > 0.5 \tag{7.2}$$

In general we can approximate this event by assuming an independent Bernoulli trial with probability q. Each time the station tries to send a data packet, the packet will be successful with probability q. This can be approximated by a geometric distribution function whose expected value is $E[Y] = \frac{1-q}{q}$. Recall that a geometric random variable counts the number of failures before a success occurs. Thus, for the worst possible receiver we have $q = 1/2$ and $E[Y] = 1$. We are now ready to prove the following theorem.

Theorem 15: *Consider a system where each station has at most $d_{max} > 1$ one-hop neighbors. Then the average channel delay experienced by a station trying to deliver a data packet can be upper bounded by*

$$D \leq 5T(n, d_{max}) \tag{7.3}$$

where $T(n, d_{max})$ is the maximum possible duration of a CRI.

Proof: To derive Eq. (7.3) we assume that for our station n_r the intended receiver is always the worst possible receiver that we can think of. Thus, each time the station n_r tries to transmit a packet it fails on average half of the time. According to our model this is true if within $T(n, d_{max})$ the station does not hear an RTR. After a failed transmission the station must return to its channel and switch from the SEND-CONTENTION state to the RECEIVE-CONTENTION state initiating a new CRI as a receiver. After the CRI ends the station transitions back to the SEND-CONTENTION state and tries once more to acquire the floor on the channel assigned to its intended receiver.

Now, since each failed transmission has a total cost of $T(n, d_{max})$ (the time waiting for an RTR) plus $E[R]$, the expected duration of the CRI (in which the station is a receiver), the expected duration of a CRI can be upper bounded as follows

$$E[R] \leq \sum_{k=0}^{d_{max}} p(k)T(n,k) \leq T(n, d_{max}) \tag{7.4}$$

where $p(k)$ is the probability of having a CRI with k stations competing for the floor, and $T(n,k)$ is the corresponding duration to resolve all k initial collisions.

The same procedure is repeated each time the station fails to deliver its data packet and must retry after one CRI as a receiver. We can approximate the retry event by assuming an independent Bernoulli trial with probability q. Each time the station tries to send a data packet, the packet will be successful with probability q. Therefore, it can be represented by a geometric distribution function whose expected value is

$$E[Y] = \sum_{y=0}^{\infty} y(1-q)^y q = \frac{1-q}{q} \tag{7.5}$$

which is equivalent to the expected number of failures before a success. Thus, the average time spend by station n_r transmitting a data packet can be written as

$$\begin{aligned} T_{xmit} &\leq \sum_{y=0}^{\infty}(1-q)^y q(y(T(n, d_{max}) + E[R]) + 2T(n, d_{max}) \\ &= (T(n, d_{max}) + E[R])E[Y] + 2T(n, d_{max}) \\ &\leq 2T(n, d_{max})(1 + E[Y]) \end{aligned} \tag{7.6}$$

Now observe that the data packet had to be originated in the last CRI in which the station was in the RECEIVER-CONTENTION state. If this was not the case the station would have remained in the RECEIVER-CONTENTION state and not switched to the SENDER-CONTENTION state. Therefore, we need to add to T_{xmit} the expected duration of the last CRI (i.e., $E[R] \leq T(n, d_{max})$) before the station switched from the RECEIVER-CONTENTION state to the SENDER-CONTENTION state. The thesis then follows from the fact that for the worst possible receiver we have $q = 0.5$ and $E[Y] = 1$. ∎

In the next result we derive a lower bound on the average channel utilization.

Theorem 16: *Consider a system where each station has at most $d_{max} > 1$ one-hop neighbors. Then the average channel utilization is bounded by*

$$U \geq \frac{(E[X] + E[Y] + 1)E[k]\delta}{E[X]E[R] + E[R] + (T(n, d_{max}) + E[R])E[Y] + 2T(n, d_{max})} \tag{7.7}$$

where $E[X]$ is the number of CRI in which station n_r remains in the RECEIVER-CONTENTION state before it leaves its channel to transmit a data packet of its own on another channel; $E[Y]$ is the number of attempts that station n_r must make before transmitting its data packet; $E[k]$ is the expected number of data packets received by station n_r per CRI; $E[R]$ is the average duration of one CRI as a receiver; $T(n, d_{max})$ is the maximum duration of a CRI and δ is the size of one data packet.

Proof: The frequency with which station n_r tries to send a data packet is determined by its packet data rate. Once again we can approximate this procedure with a geometric distribution function. At the end of each CRI a coin with probability q is tossed to decide whether or not during the previous CRI station n_r had a packet to send and thus must switch from the RECEIVER-CONTENTION state to the SEND-CONTENTION state. Hence, the expected number of CRIs occurring before station n_r has a packet to send is

$$E[X] = \frac{1-q}{q} \tag{7.8}$$

Now, let k be the number of packets received during a CRI, $p(k)$ be the probability of having a CRI with k initial collisions and $T(n, k)$ be the time needed to resolve all k collisions. It is not difficult to see that the expected duration of a CRI can be upper bounded by

$$E[R] \leq \sum_{k=0}^{d_{max}} p(k)T(n, k) \tag{7.9}$$

and that the expected number of packets received in a CRI is

$$E[k] = \sum_{k=0}^{d_{max}} p(k)k. \tag{7.10}$$

Since each station has at most d_{max} one-hop neighbors, we have that $k \leq d_{max}$. Observe that the total duration T_{total} in a channel is composed of receiving periods and transmitting periods that appear as idle periods in the channel. Furthermore, since the number of receiving periods before station n_r has a data packet to transmit on another channel is determined by the expected value for a geometric random variable, i.e., $E[X]$, it follows that the expected arrival time for a packet is $E[R]E[X]$. Once the packet has arrived, station n_r finishes as receiver the current CRI adding one more $E[R]$ period to the total duration. Then it switches to the SENDER-CONTENTION state.

Now, station n_r makes several trials before it is able to deliver its data packet. As explained in Theorem 15, the trials are also governed by a geometric distribution. Each failure has a cost of $T(n, d_{max})$ plus $E[R]$, and the number of failures before the data packet can be delivered successfully is $E[Y]$. Once station n_r hears an RTR in the channel of the intended receiver, it has at most $2T(n, d_{max})$ to content successfully and deliver its packet. Therefore, the total period in the channel can be written as

$$T_{total} \leq E[X]E[R] + E[R] + (T(n, d_{max}) + E[R])E[Y] + 2T(n, d_{max}) \qquad (7.11)$$

where $E[X]$ is the number of CRI in between two transmissions, $E[R]$ is the expected duration of each CRI and $E[k]$ is the expected number of data packets transmitted in channel ch_r per CRI. On average the amount of time that channel ch_r is being used to transmit data packets is

$$(E[X] + E[Y] + 1)E[k]\delta \qquad (7.12)$$

The thesis then immediately follows by dividing Eq. (7.12) by Eq. (7.11). ∎

7.4 Simulation

In order to verify the analytical results obtained in the previous section, simulations with different loads were performed. For the experiments we used a total of 100 stations each

surrounded by at most $d_{max} = 4$ neighbors. Therefore, the maximum number of channels was set to $n = d_{max}^2 - d_{max} - 2 = 10$. Packets were created at each station using independent Poisson generators. The simulations were repeated a number of times running on average one hour per trial. We assumed a high-speed network ($1Mbps$) in which Ethernet data packets (512 bytes) are transmitted. Furthermore, we have used the same distance between stations and defined the diameter of the network to be 1 mile. Under these assumptions, the propagation delay of the channels is $5.4\mu s$. In order to accommodate the use of IP addresses for destination and source, the minimum size of RTSs and CTSs was chosen to be 20 bytes. The RTR control packet was assumed to be 10 bytes. The maximum expected theoretical packet delay was $5T(n, d_{max}) = 5T(10, 4) \leq 98ms$.

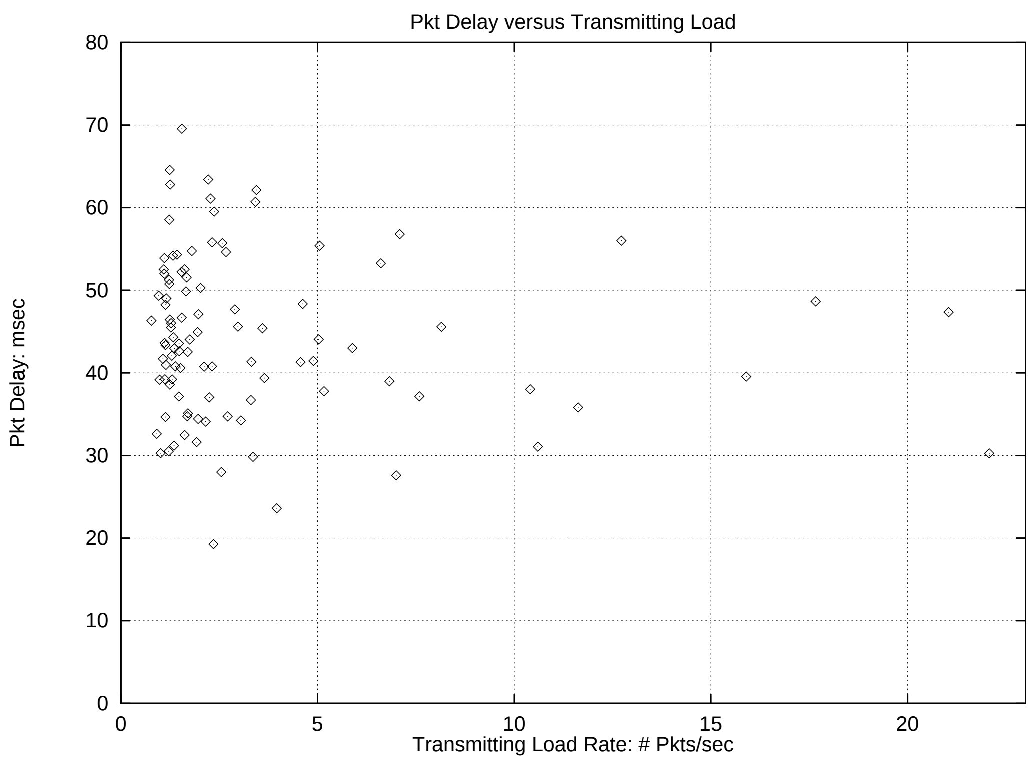

Figure 7.4: Channel delay for CARMA-MC

Figure 7.4 shows packet delay as a function of the number of data packets being transmitted per second on a channel. The load applies to the receiving channel, i.e., the

number of packets that the receiver receives per second. The delay is measure in msec and is the interval of time from the arrival of the packet to its queue until the packet was successfully delivered. The packet delay for each individual trial was always below the theoretical upper bound predicted by Eq. (7.3).

In figure 7.5 we have compared the utilization derived from our analysis to that obtained in the simulation as a function of the receivers load. To obtain the utilization of the channel we counted the intervals of time when data packets were transmitted on the channel divided by the total simulation time. The simulation results validate our analytical results.

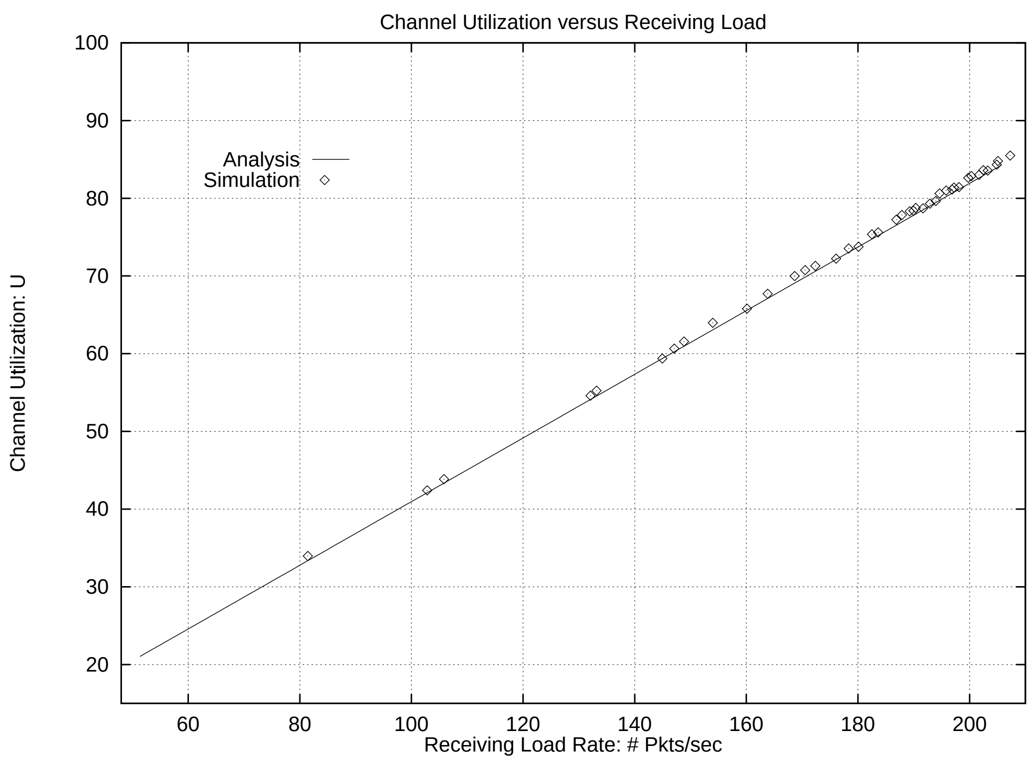

Figure 7.5: Channel utilization for CARMA-MC

7.5 Summary

CARMA-MC implements a collision avoidance and resolution protocol for wireless multichannel network. CARMA-MC is a receiver initiated protocol that is based on a deterministic tree-splitting algorithm. CARMA-MC resolves three mayor sources of packet failure. First, busy interference is resolved since the intended receiver initiates the communication and remains as a receiver until the end of the CRI. A station will never send a data packet to a receiver that is not ready to receive it. Second, co-channel interference is not an issue for CARMA-MC since no station two hops away will have the same receiving frequency. Third, contention interference is resolved by using the deterministic tree-splitting algorithm. CARMA-MC is ideal for wireless systems with fix radios, i.e., the network devices in the Metricom network.

The average channel delay is bounded and is a function of the maximum number of one-hop neighbors. Our simulation validate the simplifying assumptions that we made on our analysis.

8. SPECTRUM ETIQUETTE

In the U.S., the Federal Communications Commission (FCC) made available 6.2 GHz of spectrum and established technical rules that permit the introduction and development of communications technologies in the millimeter wave frequency bands above 40 GHz. Europe and Japan are also considering commercial uses of millimeter wave technology.

The term "millimeter wave" is taken from the fact that the wavelength of radio signals between 30 GHz and 300 GHz ranges from 10 millimeters down to 1 millimeter. The FCC action makes available three frequency bands: 47.6-47.8 GHz, 59-64 GHz, and 76-77 GHz, for unlicensed vehicle radar systems and general purpose unlicensed devices. The 59-64 GHz band was set aside as a general unlicensed band. This is an unprecedented decision in terms of bandwidth being made available and the lack of regulatory constraints.

The 59-64 GHz band could be used for wide bandwidth computer communication over point-to-point wireless links at data speeds that may exceed 5 Gbps. This would extend the data rates currently available to a fixed user through fiber optic cable. However, equipment may not be operated on this band, until an etiquette has been defined for its use. In this context, an etiquette is a specific set of access rules that permits multiple systems from different manufacturers to share the bandwidth without undue interference, and without requiring the manufacturers to adhere to the same medium-access control (MAC) or physical-layer (PHY) communication protocols. Several companies, including Hewlett-Packard, Apple, Sun, Motorola, Hughes Research, Eaton Division of Cutler-Hammer, Rockwell International, and Metricom among others, have begun defining a spectrum etiquette for sharing this band. The main challenge in the definition of such an etiquette is the huge bandwidth that is being made available, which precludes systems from listening over the entire band to try to prevent interference. According to the FCC regulations, within the 59-64 GHz band, the power density of any emission shall not exceed $9\mu W/cm^2$ at a distance of 3 meters. The power density of any emissions outside the 59-64 GHz band must consist solely of spurious emissions and must not exceed $90pW/cm^2$ at a distance of 3 meters. The power

measurements will be average measurements based on a 1 MHz bandwidth. Within this constraints, the etiquette should fulfill the following requirements:

- The etiquette should provide a substantial reduction in the probability of interference between co-existing systems.
- The etiquette should seek to promote realization of high-speed communications while attempting not to fore-close low speed communications.
- The etiquette should be flexible enough to allow as many applications as possible to effectively co-exist in the band.
- The etiquette should not have a major negative impact on the economic feasibility of systems.
- The etiquette should provide for the diverse needs of both continuous-connection and burst-mode systems.
- In all portions of the band where etiquette applies, only one etiquette should be used.
- The etiquette must be kept simple. To this end, effectiveness may be traded off for simplicity. The etiquette must use as few layers as possible in the standard OSI stack.
- The etiquette must promote efficient use of the spectrum.
- The etiquette must be open and non-proprietary, it must have openly-available set of procedures.

In this chapter, we propose a listen-before-transmit etiquette for heterogeneous systems implementing different PHY and MAC protocols and based on power sensing over a control channel used to schedule access to the rest of the band, which is partitioned into data channels. Each system consists of any set of nodes using the same PHY and MAC protocols, and two nodes from different systems cannot decode one another's transmissions in any data channel of the band. Each data channel is meant to be used by an individual system (i.e., two or more nodes using the same PHY and MAC layers) on long-term and persistent basis. The control channel is used to exchange information about the band use activity in the area. Prospective transmitters listen to the control channel to get information about the data channels occupancy; this eliminates the need to listen to the entire wide band.

The only means by which systems can exchange information with one another over the control channel is the duration of each others' transmissions, which are perceived only as noise, and no information is exchanged across systems over the data channels defined in the band. A novel transmission encoding is defined based on this basic control-channel feedback that allows systems to ascertain which system can use which data channel at which time, without interference.

The proposed etiquette is meant for assignment of data channels to systems, rather than individual stations. Hence, it makes sense to have the data channels be of substantial width such as, for example, to support OC3 rates. For the 59-64 band, this means that the number of data channels is around 30.

The chapter is organized as follows. Section 8.1 describes the assumptions that we make for our model. Section 8.2 gives a detailed description of the control channel and the proposed etiquette basic operation. Section 8.3 presents analytical and simulation results showing that the proposed etiquette is fair for all the co-existing systems, fully utilizes the spectrum, provides bounded delays for data-channel acquisition time by any given system, and provides minimum channel-use guarantees. Our analysis shows that the etiquette's use of the available band approximates that of an optimal assignment of data channels to co-existing systems. Section 8.4 offers our concluding remarks on the propose etiquette.

8.1 Definitions and Assumptions

Throughout this chapter, the *band* for which the proposed etiquette is used is the 59-64 GHz general unlicensed band. In our model, the band is divided into s *data channels* (i.e., channels that are used to transmit data once the channel has been acquired) and a *control channel* (i.e., a small predefined portion of the band for the exchange of scheduling information among the systems). In our model, a *system* is a collection of nodes sharing the same PHY- and MAC-layer protocols. We make no assumptions on the way in which a system allocates the data channels to its nodes or schedules their use. However, we adopt a system-level approach in the operation of the etiquette. More specifically, we assume that a

single station in a system is in charge of participating in the etiquette activity in the control channel to obtain usage rights for data channels. We refer to this station as the *etiquette designated station (EDS)*. We note that the EDS for a system need not always be the same i.e., a system can change its EDS at will. The idea of an EDS is to ensure that every system is represented only once, so that stations from the same system do not compete in the control channel. The EDS is also in charge of notifying the rest of the stations in its own system about availability of data channels. The EDS is the only node in the system that is allowed to send reservation signals in the control channel, all other nodes in the system are only allowed to send an *echo*, i.e., a response to such a reservation signal.

We assume that no information is exchanged among the systems over any of the data channels. Each system is fully independent in that its PHY- and MAC-layer protocols which can be completely different from any other system's PHY/MAC layer.

A unique identifier (ID) is assigned to each EDS, which in practice could consist of three fields: the device's FCC ID number; the device's serial number; and a user-definable field.

The control channel is organized in *frames*, each of which is further divided into *periods* made up of several slots; the exact structure of these is further discussed below. The number s of data channels is assumed to be predefined, and that number determines the length of a frame in the control channel.

8.2 Etiquette Description

In general, the etiquette described in this chapter can be defined as a specific set of access rules that permits multiple systems with different PHY and MAC protocols to compete for channel usage one channel at a time on the 59-64 GHz band.

A system cannot compete for a specific channel but rather for whichever channel happens to be available next. A system cannot target a specific channel but must use whichever channel it happens to acquire. The allocation of channels is done in order and a system cannot compete for a second data channel, until all other systems acquire their first data channel. A system may not be removed from its current data channel if the number of

systems is smaller than the number of data channels. If there are more systems than data channels, then the system may be removed from the data channel it has acquired.

Competition for a data channel is resolved by way of a collision-resolution algorithm. This algorithm requires that the activity on a channel (idle, success, collision) be known to the sender, but the sender cannot determine this by itself. This is why we introduce the echo mechanism; the EDS of a system transmits a control packet in the control-channel period that can be understood only by other stations of the same system, which can provide an echo; other systems perceive this as noise, while the EDS understands the signal.

Based on the channel organization assumptions described above, the proposed etiquette operates over a control channel organized into *frames*. The etiquette consists of framing mechanisms and mechanisms for the reservation of data channels. All this is done without having the system share a common PHY-layer protocol.

Framing is accomplished using the time of transmission as the only feedback to systems. A frame in the control channel consists of a framing signal followed by a sequence of channel-control periods. There is one channel-control period for each data channel, and there is a unique predefined association between a channel-control period and a data channel. The framing signal consists of a transmission pattern guaranteed to differ from any pattern within and across boundaries of channel-control periods.

The signaling used in each control period is based on the notion that different systems can only detect signal duration from one another (perceived as noise), while stations in the same system can actually exchange data. Thus, the activity on the control channel will be in the form of request-echo pairs. The EDS of a system will transmit a certain information packet in the control-channel period which will be understood only by other stations of the same system, who could provide and echo. Other systems will perceive this as noise. Of course, if more than a single EDS transmit concurrently a collision occurs and everybody perceives this as noise.

Data channels are reserved by means of requests made dynamically by the EDS. Because such requests are made when stations in a system require a data channel, requests from

different systems may occur at the same time. Although such collisions could be resolved using a random backoff approach similar to what simple MAC protocols do (e.g., ALOHA, CSMA), an etiquette based on such an approach would not be stable and could not guarantee a maximum delay for a system to acquire a data channel. Because of the desirability of providing channel-assignment delay guarantees, we designed our etiquette using a deterministic tree-splitting collision resolution algorithm [9] tailored for the case in which the number of systems competing for the available data channels is finite.

Fig. 8.1 shows an example of an etiquette frame with three channel-control periods suitable for a band with three data channels. Let τ be the maximum propagation time for systems in the band, and let γ be the duration of a reservation request (as well as the length of an echo signal). We define $\delta = \gamma + \tau$, which is the time required for a transmitted signal to be received. As was indicated earlier and as the Fig. 8.1 illustrates, the framing signal and channel-control periods consists of slots of duration δ.

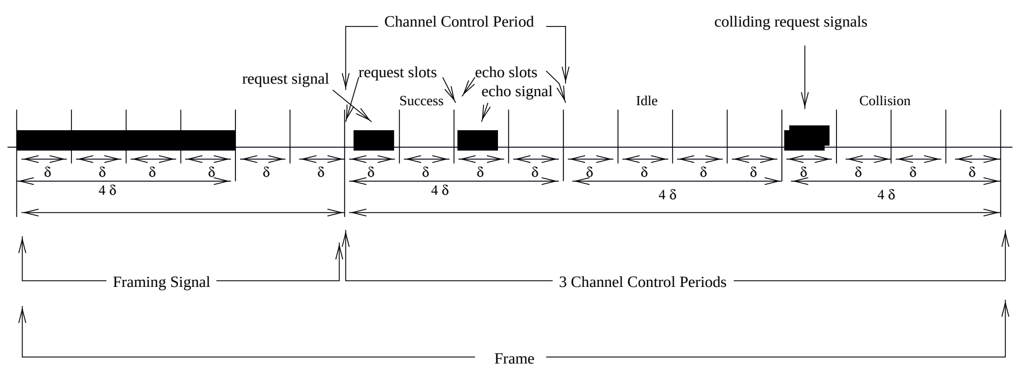

Figure 8.1: An example of an etiquette frame with three channel-control periods.

Because the collision resolution algorithm used to resolve channel requests can take multiple frames, the framing signal at the beginning of each frame must also specify whether or not an unfinished round of collision resolution is taking place in the present frame. This informs EDS with new channel requests to wait until the current round of requests is satisfied. In the proposed etiquette, a framing signal consisting of four consecutive slots with jamming (transmission of a signal by any of the systems), followed by two idle slots indicates the beginning of a *new* collision-resolution round; This is the case shown in Fig. 8.1.

A framing signal consisting of three jamming slots followed by two idle slots indicates the *continuation* of a collision-resolution round.

Each channel-control period consists of four slots. These four slots are encoded to inform all systems of the availability, assignment, or ongoing contention for the corresponding data channel. The first two slots of the channel-control period are control slots used by systems to indicate current ownership of, or a request for the corresponding data channel. The last two slots of a channel-control period are echo slots used to provide feedback.

8.2.1 Initializing Frames

Let s be the number of data channels defined on the band. If a system wishes to use one of the data channels its EDS first listens to the activity on the control channel. If the channel is idle for at least $4s\delta$ seconds (where s is the number of data channels in the band) plus the duration of a framing signal, the EDS will send a framing signal of duration 4δ followed by two idle slots (2δ). This determines the beginning of a new reservation frame as well as a new collision-resolution round. Each EDS requesting the use of a channel sends a request signal of size γ within the first slot of the four slots associated with the respective channel-control period. The request signal is followed by an idle period of size δ. If the request was the only one, the intended receiver within the same system will understand the signal and send an echo signal back. For all the other systems in the network this request signal is noise, therefore, they will not send an echo signal.

We denote by $< abcd >$ the encoding used in any given channel-control period, where a,b,c and d are 0 or 1, depending on whether the corresponding slot is silent (idle) or there is a signal from at least one system. In terms of this notation, a station must listen for $< 111100 >$ to detect a new frame that does not have an ongoing round of request resolution. The signal $< 11100 >$ indicates that the frame has an ongoing resolution round and the station must refrain from requesting a channel.

Note that because any two EDSs are within τ seconds of one another, and because the encoding used in the channel-control periods prohibits four or three consecutive jamming

slots, all active EDSs detect the beginning of the frame at the end of last jamming slot sent by any such station for framing purposes. Also note that a station must listen for an entire frame duration before attempting to request any channel. This implies that a station knows the state of the band assignment when it makes its request.

8.2.2 Signaling in the Channel-Control Period

Each station requesting the use of the channel sends a request signal of size γ within the first slot of the four slots assigned to the respective channel-control period. The request signal is always followed by an idle period of size δ. If only one system requested the channel, one or multiple receivers within the same system understand the signal and one of them send an echo signal back in the third slot of the channel-control period. For all the other systems, the request signal appears as noise for which they do not send an echo signal. This case corresponds, therefore, to an encoding of $< 1010 >$ in the channel-control period of the channel. If the request was unsuccessful due to a collision of multiple requests, no station will understand the request and consequently none will respond in the echo slot. This results in a channel control period of $< 1000 >$. A channel-control period with an empty signal $< 0000 >$ corresponds to an unused data channel.

We have seen that a successful request for a channel is encoded by $< 1010 >$, an unsuccessful request for the channel is $< 1000 >$ and an empty channel by $< 0000 >$. It is clear that, because a station in a system needs feedback within its own system, a period must include at least two slots, one for a request and one for the echo to the request. The reason why four slots are used to encode each channel-control period is that a system must also convey the following additional information:

- The code $< 0010 >$ is used to signal that the corresponding data channel is busy but was not the last data channel in the band to be reserved. This is important in the collision resolution algorithm as described later.
- The code $< 0001 >$ signals that the corresponding data channel is busy and it was the last data channel in the band to be reserved. Other EDSs will attempt to reserve

data channels starting with the next data channel in the band. This code tells EDSs wishing to request a channel to start their bids on the next channel. As soon as a new last data channel is reserved the EDS changes the code from $< 0001 >$ to $< 0010 >$ in the next frame.

In addition, because of the need to guarantee uniqueness of the framing signal, no sequence of channel-control periods may contain a code of $< 1111 >$, $< 1110 >$, or $< 0111 >$ which are prefixes of framing signals. This is achieved by having the second slot of a channel-control period always be 0.

Once a system acquires a channel it can keep using it as long as it needs it or until it is challenged by another system. A system that has acquired a channel must send an echo signal in each subsequent frame to ensure the continuing use of such channel. If no echo signal is sent other systems are free to make use of the data channel. If the current system is challenged by a new system, the new system sends a request signal. The old system notices the request signal, i.e., it is aware of a signal in the request slot, and refrains from emitting an echo signal. Since the new system understands the request signal and such signal was not interfered by the old system, an echo signal reserving the channel is transmitted. If two or more systems send a request signal within the same slot, none of the systems will understand it, therefore, no echo will be sent. Each system involved in the collision will select a new channel among the free channels. If all the channels are busy, the system must select randomly among one of the busy channel-control periods.

8.2.3 Resolving Channel Requests Conflicts

Because a single EDS in each system interacts with EDSs from other systems, we describe how channel requests are resolved by referring to a system making the request, rather than the EDSs.

In the proposed etiquette, a system that requires access to a data channel listens to the control channel for an entire frame to ascertain the state of the band, i.e., which data

channels are free, whether there is an ongoing resolution of channel requests, and which was the last data channel to be assigned.

Each system is assigned a unique identifier, and maintains a stack, and two variables ($LowID$ and $HiID$). $LowID$ is initially the lowest ID and $HiID$ the highest ID given to any system. Together, they constitute the allowed ID interval that can attempt to reserve a channel. If the ID of a system is not within the allowed interval, it cannot request a channel. The stack is simply a storage mechanism for ID intervals that are waiting to get permission to request a channel.

Initially, all channels are free and there is no activity in the control channel or in the channels corresponding to each channel-control period.

When a passive system requires a channel, it first listens to the control channel. When the framing signal is detected, it listens for the entire frame, and records the state of each of the s channels. If the control channel is idle (i.e., no framing signal is found) for a period equivalent to the size of a frame ($4s\delta + 4\delta + 2\delta$ seconds) the system transmits a framing signal. The framing signal determines the beginning of the frame.

When a system attempting to reserve a channel detects that there is an unfinished round of channel request resolution, which is detected when the framing signal is $< 11100 >$, the system waits until it reads an entire frame starting with a framing signal $< 111100 >$ indicating that the prior channel allocation requests have been resolved. Starting with the first channel-control period, all systems wishing to acquire a channel transmit a request signal in the first slot (the request slot) of the first channel-control period leaving the next slot idle. The sender then waits and listens to the channel for one slot for an echo signal. An echo signal is transmitted by one or multiple stations in the same system only if the request signal is heard free from errors. This is the case if the system is the only one that transmitted a request signal. If noise is detected in the request slot the echo slot is left idle. An empty echo slot is interpreted by the requesting systems as a collision of channel requests.

If an echo signal is received, the system acquires the channel and begins transmitting its

data in the corresponding data channel. The system has unlimited use of the band, until it is challenged by another system or until it does not need the channel anymore, after which the system releases the channel by stopping the transmission of request and echo signals. As long as a system maintains access to a data channel, it transmits the code $< 0010 >$ in the corresponding control period of each frame if the data channel was not the last channel assigned during the last resolution round, and transmits the code $< 0001 >$ otherwise. This permits all systems that need access to a new data channel to begin their requests with the next unused data channel following the data channel with a control period having a code of $< 0001 >$. An unused data channel is one for which its control period had a code of $< 0000 >$ (empty) or $< 1000 >$ (a collision of two or more requests and no current system in the channel) in the previous frame.

If the sender of a request signal does not receive an echo during the echo slot, the sender and all other systems participating in the etiquette know that a collision of requests has occurred. As soon as the first collision takes place, every system divides the ID interval $(LowID, HiID)$ into two ID intervals. The first ID interval is $(LowID, LowID + \lceil\frac{HiID+LowID}{2}\rceil - 1)$, which we will call the backoff ID interval, while the second ID interval is $(LowID + \lceil\frac{HiID+LowID}{2}\rceil, HiID)$ and is called the allowed ID interval. Each system updates the stack by executing a PUSH stack command, where the key being pushed is the backoff ID interval. After this is done, the system updates $LowID$ and $HiID$ with the values from the allowed ID interval. This procedure is repeated each time a collision is detected.

Only those systems that were involved in the first collision are allowed into the collision-resolution phase. All other systems are in REMOTE state and simply keep track of the state for each channel, as well as the allowed ID interval and the backoff ID interval. A system remain in REMOTE state remains in this state until all collisions are resolved from the previous round.

Collision resolution of requests evolves in terms of collision-resolution intervals, of which there are three cases: idle (i.e., code $< 0000 >$), success (i.e., code $< 1010 >$), or collision interval (i.e., code $< 1000 >$). In the first interval of the collision-resolution phase all

systems in the allowed ID interval that are in the REQUEST state try to retransmit a request signal. If none of the systems within this ID interval request the channel (i.e., code $< 0000 >$), a new update of the stack and of the variables $LowID$ and $HiID$ is due. Each system executes a POP command in the stack. This new ID interval now becomes the new $HiID$ and $LowID$. The same procedure takes place if, during the first collision-resolution interval, only one system is requesting the channel; the originator receives the echo signal (i.e., the code $< 1010 >$ occurs) and the system begins transmission in the assigned channel. The third case of a collision-resolution interval is for multiple systems to request the same channel, causing a collision (i.e., code $< 1000 >$). The systems in the allowed ID interval are once more split into two new ID intervals and the stack as well as the variables for each system is updated.

The etiquette repeats the above steps, until all the requests have been resolved. Notice that, as soon as the backoff stack becomes empty and there are no values in the allowed interval, all systems know that all the collisions of channel requests have been resolved for the of requests resolution and a new round can start, if there are systems that require data channels.

8.2.4 Example of the Etiquette's Operation

We illustrate the etiquette's operation using a simple example (see Fig. 8.2) with four systems labeled n_{00}, n_{01}, n_{10}, and n_{11}, and three channel-control periods per frame, labeled $s1$, $s2$ and $s3$. The framing signals (i.e., $< 111100 >$ and $< 11100 >$) are omitted for simplicity and we consider one round of collision resolution. We also assume that once a data channel is busy it remains busy until it is challenge by another system. Because the example assumes the beginning of a new request-resolution round, the backoff stack is empty and the allowed ID interval contains all the systems in the network, i.e., the allowed ID interval is (n_{00}, n_{11}) (Step 0 in Fig. 8.2).

At the end of the previous request-resolution round in the example, systems n_{11} and n_{10} have acquired channels $s3$ and $s1$ respectively (Step 0 in Fig. 8.2). Based on the state of

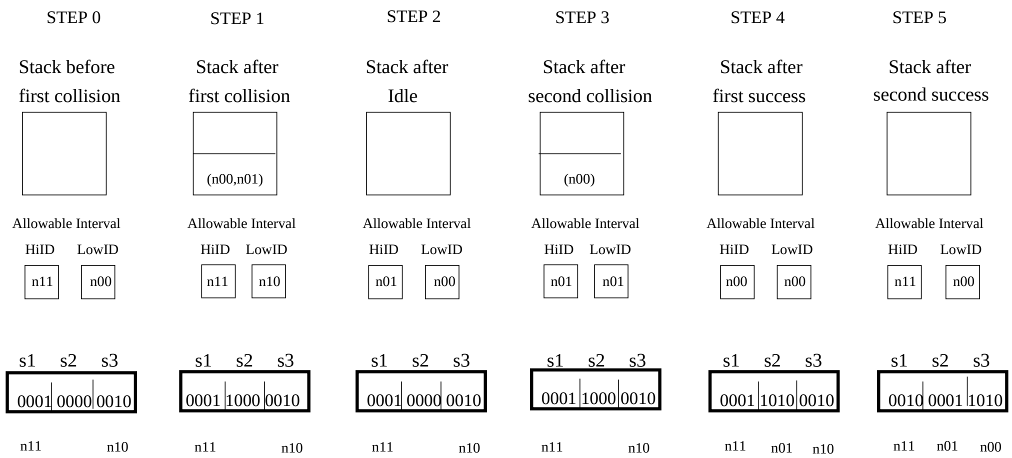

Figure 8.2: An etiquette's operation for four systems with three channel-control periods. The state of the last round is illustrated in step 0; systems n_{11} and n_{10} have acquired two out of the three available channels; systems n_{00} and n_{01} request a channel in the next round.

the channels, all systems know that the channel $s1$ was the last data channel in the band to be reserved (code $< 0010 >$), while channel $s3$ was busy but was not the last data channel in the band to be reserved (code $< 0010 >$). If all channels are busy, the next contention channel is the next data channel after the channel with code $< 0001 >$. On the other hand, if there are data channels that are not currently busy (code $< 0000 >$) or with an unsuccessful request (code $< 1000 >$), the next contention will be done in the next empty or unsuccessful request channel following the data channel with code $< 0001 >$. Therefore, the next contention in our example occurs in channel $s2$.

After the framing signal $< 111100 >$ is transmitted by all the active participating systems, all systems notice the beginning of a new request-resolution round. System n_{11} continues sending the echo signal in $s1$ (code $< 0001 >$), indicating that the channel is still in use. Systems n_{00} and n_{01} use channel $s2$ to request a data channel since both are in the allowed ID interval. Although systems n_{10} and n_{11} are within the allowed interval they do not participate in the contention of channel $s2$ because they already have acquired a data channel. The first collision in channel $s2$ occurs (Step 1 in Fig. 8.2) with systems

n_{00} and n_{01} each sending a request signal. If the request was unsuccessful due to a collision of multiple request, no feedback exists (i.e., corresponding to code $< 1000 >$); the backup stack and the allowed ID interval are updated. Systems n_{00} and n_{01} are members of the backoff ID interval; therefore, they both are on hold, they must wait until the collisions in the allowed ID interval are resolved. In the next frame, systems n_{10} and n_{11} are allowed to request data channel $s2$. Finally, the unsuccessful request in channel $s2$ is followed by an echo signal from system n_{10} (code $< 0010 >$) in channel $s3$, terminating the first frame.

The second frame is initiated with the signal $< 11100 >$ transmitted by all the active participating systems. System n_{11} continues sending the echo signal in the first channel-control period $s1$ (code $< 0001 >$). In the next channel-control period, $s2$, an idle period occurs (code $< 0000 >$, see Step 2 in Fig. 8.2), because systems n_{10} and n_{11} are in the allowed ID interval but do not need to request the channel. At the end of the channel-control period, all systems notice that the code was $< 0000 >$, which means that there were no collisions; accordingly, the systems in the system must update their intervals and the stack. They execute a POP-stack command and the new allowed interval is (n_{00}, n_{01}) (Step 2 in Fig. 8.2). The idle channel-control period for channel $s2$ is followed by an echo signal from system n_{10} (code $< 0010 >$) in the $s3$ channel-control period, terminating the second frame.

The third frame is initiated with the framing signal $< 11100 >$ transmitted by all the active participating systems. System n_{11} continues sending the echo signal in the first channel-control period $s1$ (code $< 0001 >$). In the second channel-control period both systems n_{00} and n_{01} transmit an echo signal (Step 3 in Fig. 8.2) and another collision occurs. Because a collision occurred, the allowed ID interval is split, i.e., system n_{01} is within the allowed interval while the n_{00} system must wait, its interval is the top of the stack. The third frame terminates with system n_{10} sending the echo signal in the third channel-control period $s3$.

The fourth frame is initiated with the signal $< 11100 >$ transmitted by all the active participating systems. System n_{11} continues sending the echo signal in the first channel-

control period $s1$ (code $< 0001 >$). Since in the second channel-control period only one system is is in the allowed ID interval, system n_{01} acquires channel $s2$ (code $< 1010 >$ in Step 4 in Fig. 8.2). At the end of the channel-control period the systems do an update, i.e., a POP-stack command. System n_{00} is the new allowed ID interval and the backup stack is empty. The fourth frame terminates with system n_{10} sending an echo signal in the third channel-control period $s3$. All three channels are busy, therefore, the next contention is done in the channel-control period following the code $< 1010 >$. In the example the contention is continued in $s3$ because $s2$ was the last busy data channel to be reserved.

The fifth frame is initiated with the framing signal $< 11100 >$. System n_{11} continues sending the echo signal in the first channel-control period $s1$. Because $s1$ is no longer the last data channel to be reserved the code $< 0001 >$ is replaced by the code $< 0010 >$. In the second channel-control period system n_{01} sends the echo signal $< 0001 >$ instead of code $< 1010 >$. In the third channel-control period System n_{00} can request and acquire data channel (Step 5 in Fig. 8.2). At the end of the third channel-control period the systems do an update, i.e., a POP-stack command. Both the backup stack and the allowed ID interval are empty. The termination of the collision-resolution phase is determined by an empty stack and an empty allowed ID interval. The systems empty their stacks and update the allowed ID interval permitting all systems to contend in the next request-resolution round.

8.3 Etiquette Performance

In this section we show that the performance of the proposed etiquette approaches that of an optimal assignment of channels to systems from the standpoint of data channel utilization. We obtain a lower bound on the etiquette's throughput. We begin our analysis by finding the average number of steps required until m channel request are resolved. A step is define as a channel control period (idle, success, collision) and has the length of 4δ. We then derive the throughput of any given system.

8.3.1 Average Number of Request-Resolution Steps

Let there be n systems in the network, each with a distinct ID and $m \leq n$ of the systems request one data channel each. The total number of data channels available is s and are assume empty, i.e., unused. All m systems compete for the first channel and sequentially continue in the next data-control period as describe in the example in Section 8.2.4, until all the m requests are resolved.

Theorem 17: *Let there be $m > 1$ requests for channel assignment from m distinct systems (one request per system) and let there be $n >= m$ maximum number total systems, then the average number of steps required until all m channels requests are resolved is*

$$\overline{\mathcal{T}}(n,m) = \sum_{i=\mu}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{\mathcal{T}}\left(\alpha, m-i\right) + \overline{\mathcal{T}}\left(\beta, i\right) + 1\right] \tag{8.1}$$

where

$$\begin{aligned}
\alpha &= \lceil n/2 \rceil; \;\; \beta = n - \alpha = n - \lceil n/2 \rceil \\
\mu &= \begin{cases} 0 & \textit{if } m \leq \alpha \\ m - \alpha & \textit{if } m > \alpha \end{cases} \\
\nu &= \begin{cases} m & \textit{if } m \leq \beta \\ \beta & \textit{if } m > \beta \end{cases}
\end{aligned}$$

Proof: We define a step as a channel control period of size 4δ. It is trivial that for all $n \geq 1$, $\overline{\mathcal{T}}(n,0)$ and $\overline{\mathcal{T}}(n,1)$ equal 1, i.e., we need in each of these cases one step. If we have in total two systems and both send a request signal within the same request slot, the average number of steps is $\overline{\mathcal{T}}(2,2) = 3$, one for the collision and two for the two successful request/echo exchanges.

With this initial conditions and following the tree-splitting algorithm we are in a position to find the average number of steps for $\overline{\mathcal{T}}(3,2)$. Since $m = 2$ we have to split the three total number of systems ($n = 3$) into two splits $\alpha = 2$ and $\beta = 1$ respectively. Therefore, we can either have 2 systems requesting a channel in the α-split and none in the β-split; or 1 system in the α-split and 1 in the β-split. The probability that 2 systems

requesting a channel are in the α-split while the remaining 0 requesting systems are in the β-split is given by $P\{(2 \in \alpha) \wedge (0 \in \beta)\} = \frac{\binom{2}{2}\binom{1}{0}}{\binom{3}{2}}$ and the probability that 1 systems requesting a channel is in the α-split and 1 requesting systems is in the β-split is given by $P\{(1 \in \alpha) \wedge (0 \in \beta)\} = \frac{\binom{2}{1}\binom{1}{1}}{\binom{3}{2}}$. For each of these cases the probability of the split must be multiply by the average number of steps for the right split plus the average number of steps for the left split plus one step for the root of both splits. Therefore,

$$\overline{\mathcal{T}}(3,2) \quad = \quad \sum_{i=0}^{1} \frac{\binom{2}{2-i}\binom{1}{i}}{\binom{3}{2}} \left[\overline{\mathcal{T}}\left(2, 2-i\right) + \overline{\mathcal{T}}\left(1, i\right) + 1\right] \tag{8.2}$$

We assume that, for all α and β, the average number of steps $\overline{\mathcal{T}}(\alpha, m)$ and $\overline{\mathcal{T}}(\beta, m)$ are known. If n is even, $\alpha = \beta = \frac{n}{2}$; otherwise, $\beta = \alpha - 1$. The probability that $m - i$ systems requesting a channel are in the α-split while the remaining i requesting systems are in the β-split is given by $P\{(m - i \in \alpha) \wedge (i \in \beta)\} = \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}}$. Therefore, the average number of steps for this specific split, i.e., $m - i$ systems in the α-split and i systems in the β-split is equal to the average number of steps for the α-split, plus the average number of steps for the β-split, plus one step for the root of both splits, times the probability of such a split, i.e., $\frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{\mathcal{T}}\left(\alpha, m-i\right) + \overline{\mathcal{T}}\left(\beta, i\right) + 1\right]$.

For the average number of steps, $\overline{\mathcal{T}}(n, m)$, we need to consider the cost as well as the probability of each of the possible α- and β-splits. Therefore,

$$\begin{aligned} \overline{\mathcal{T}}(n,m) \quad &= \quad \frac{\binom{\alpha}{m-\mu}\binom{\beta}{\mu}}{\binom{n}{m}} \left[\overline{\mathcal{T}}\left(\alpha, m-\mu\right) + \overline{\mathcal{T}}\left(\beta, \mu\right) + 1\right] + \ldots \\ &\ldots + \quad \frac{\binom{\alpha}{m-\nu}\binom{\beta}{\nu}}{\binom{n}{m}} \left[\overline{\mathcal{T}}\left(\alpha, m-\nu\right) + \overline{\mathcal{T}}\left(\beta, \nu\right) + 1\right] \end{aligned} \tag{8.3}$$

There are three possible μ-ν combinations. First, if $m \leq \alpha$ and $m \leq \beta$, then $\mu = 0$ and $\nu = m$. In the second case, $m \leq \alpha$ and $\beta < m$; therefore, $\mu = 0$, while $\nu = \beta$. Finally, if m is greater than both α and β, then $\mu = m - \alpha$ and $\nu = \beta$. Note that the parameter m

cannot be $> \alpha$ and $\leq \beta$ at the same time because $\beta \leq \alpha$; accordingly, this case is excluded. The sum of the average number of steps for each of the possible splits yields Eq (8.1). ∎

Theorem 18: *Starting with s empty channels and m out of the n total number systems requesting the use of a channel, the total number of channel-control periods required until the kth successful request/echo signal exchange is*

$$\overline{\mathcal{T}}(n,m,k) = \sum_{i=\mu}^{m-k} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{\mathcal{T}}(\alpha, m-i, k) + 1\right] + \sum_{i=m-k+1}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{\mathcal{T}}(\alpha, m-i, m-i) + \overline{\mathcal{T}}(\beta, i, i+k-m) + 1\right] \tag{8.4}$$

Proof: It can clearly be seen that if all k successes are within the α-split the steps in the β-split can be dropped altogether. Therefore, if we stop the recursion in Eq. (8.1) as soon as the kth successful request/echo signal exchange is achieved, than Eq. (8.1) can be rewritten as Eq. (8.4). ∎

If we set $k = 1$ in Eq. (8.4) we get the average number of steps up to the first successful request/echo exchange.

$$\overline{\mathcal{T}} \quad (n,m,1) = \sum_{i=\mu}^{m-1} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{\mathcal{T}}(\alpha, m-i, 1) + 1\right] + \sum_{i=m}^{\nu} \frac{\binom{\alpha}{m-i}\binom{\beta}{i}}{\binom{n}{m}} \left[\overline{\mathcal{T}}(\alpha, m-i, m-i) + \overline{\mathcal{T}}(\beta, i, i+1-m) + 1\right] \tag{8.5}$$

According to the etiquette, if all the s channels are busy, each of the m original systems requesting a channel contend in the same channel-control period. The collision-resolution steps are executed in each consecutive channel control period allocating channels as the resolution progresses; therefore, the number of channel control periods needed until all m systems are allocated a channel is given by Eq. (8.1).

8.3.2 Etiquette's Throughput

We define the throughput of a data channel in the band as

$$S = \frac{\bar{t}_{in}}{\bar{t}_{in} + \bar{t}_{out}} \tag{8.6}$$

where $\bar{t}_{in}$ is the average busy period for any given system, i.e., the amount of time during which the system is using the channel to transmit data. $\bar{t}_{out}$ is the average acquisition delay, i.e., the average interval between two consecutive busy periods. $\bar{t}_{in}$ can also be visualized as the average duration a system spends in a channel before it is forced to release the channel, and $\bar{t}_{out}$ is the access delay or the average duration that it takes a system to acquire a channel.

We will first assume a network with s data channels and n total number of systems, out of which m systems compete to acquire a channel. We assume that a system can at most acquire one channel at any given time. In the first part of the analysis we are interested in knowing the average number of frames required until all s channels are being use if we have m new systems trying to acquire a data channel for the case that we start with all s channels free of users.

For all $n \geq m \geq k \geq 1$, Theorem 18 determines the average number of steps $\overline{\mathcal{T}}(n, m, k)$ required for up to k successful request/echo exchanges, while Theorem 17 determines the average number of steps $\overline{\mathcal{T}}(n, m)$ required until all m collisions are resolved. Therefore, because there are s steps per frame, the average number of frames $\overline{\mathcal{F}}(n, m, s)$ required until all m systems are assigned a channel is

$$\overline{\mathcal{F}}(n, m, s) = \lceil \overline{\mathcal{T}}(n, m)/s \rceil \quad \text{if} \quad m \leq s \tag{8.7}$$

We can compare $\overline{\mathcal{F}}(n, m, s)$ to the optimal case in which all m systems are assigned m channels in exactly m steps. The optimal case assumes that there are only successful request/echo exchanges. Therefore, the total number of frames required for the optimal case is

$$\overline{\mathcal{F}}(n,m,s) = \lceil m/s \rceil \quad \text{if} \quad m \leq s \tag{8.8}$$

Fig. 8.3 shows the results for the analysis as well as the simulation. In the simulation, the total number of systems in the network (n) was set to 100 and the number of channels (s) was set to 30. Starting with s empty channels m random systems requested a data channel. For each m, 100 trials were simulated, each with m different systems requesting a data channel. For each trial we kept track of: (a) the acquisition delay, i.e., the time (measured in frames) required for each requesting system to acquire a data channel; (b) the busy period, i.e., the time a given system uses the data channel before it releases the channel; and (c) the throughput, i.e., the ratio of time the given system is busy versus the total time.

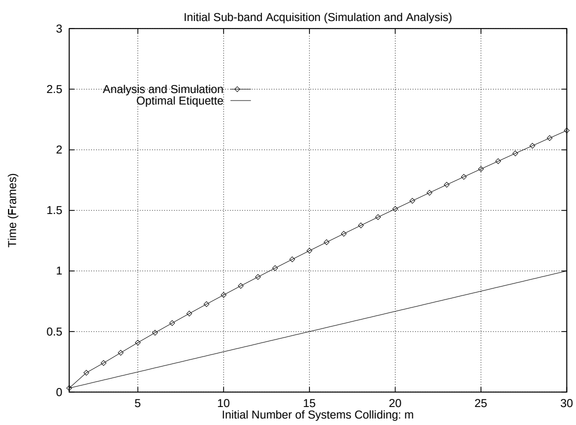

Figure 8.3: Total number of frames needed to resolve m initial collisions.

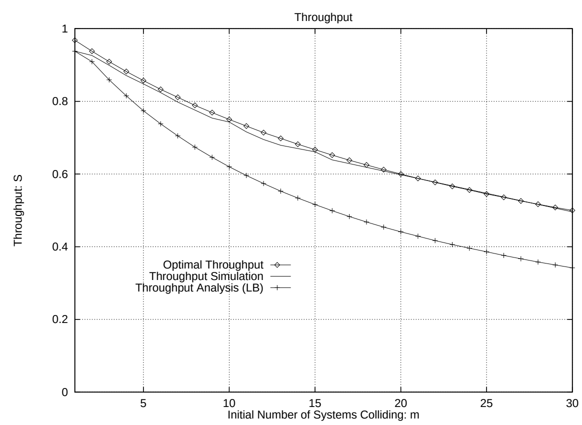

Figure 8.4: Throughput for the optimal etiquette, i.e. the upper bound, simulation, and the lower bound as a function of m initial collisions.

As shown in Fig. 8.3, the proposed etiquette requires twice the number of frames compared to the optimal etiquette. The optimal etiquette would allocate the request in a strictly linear number of steps. It is a theoretical lower bound and represents the best possible performance.

In the second part of the analysis we are interested in finding bounds on the delays for data-channel acquisition by any given system. We assume $m + s$ systems competing for s channels at any given time. Systems that loose their channel wait until the end of the current collision-resolution round and try again in the next collision-resolution round. As soon as the collision-resolution round is over all the systems that lost their channels compete for a spot. We assume that in every collision-resolution round all the s channels are being used and that m systems compete to acquire a channel, i.e., in each collision-resolution round m new systems enter replacing m old systems. The old systems contend for a channel in the next collision resolution round according to the etiquette rules.

Let x denote the number of full collision-resolution rounds from the time a system acquires a channel until it looses the channel, and k an integer from 1 to m denoting the request/echo exchange in which the given system acquires the channel. There can be at most m request/echo exchanges per collision-resolution round, and k' denotes the number of successful request/echo exchanges in the last collision-resolution round before a given system looses the channel. Let us also define $\bar{t}_{in}$ as the average time any given system uses a channel before it must release it, i.e., the average busy period. The throughput of any given system is obtained directly from the following two theorems.

Theorem 19: *For $m+s$ systems in a network the average busy period for any given system given that $m \leq s$ is*

$$\bar{t}_{in} = \frac{1}{m} \sum_{k=1}^{m} (1+x)\overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,k) + \overline{\mathcal{T}}(n,m,k') \tag{8.9}$$

Proof: Assume that all the s channels are busy and a given system acquires a channel at the kth successful request/echo exchange within a collision-resolution round of length $\overline{\mathcal{T}}(n,m)$ steps. If $m \leq s$, s successful request/echo exchanges must take place before the system entering at kth success must give up the reserved channel. This is true since the collision-resolution algorithm persists in the same channel until a success is achieve moving to the next channel-control period. Therefore, at the end of the first collision-resolution round $\overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,k)$ frames later we have hat $m-k$ new systems acquiring a channel, i.e.,

we still have $s-m+k$ systems requesting a channel before the kth system has to release the channel. Therefore, there are $x = \lceil (s+k-m-1)/m \rceil$ collision-resolution rounds in between the first round (when the given system acquired a channel) and the last collision-resolution round (when the given system looses the channel). k' is deterministic and is a function of k and s. It can be express as $k' = s + k - (1 + x)m$. Therefore, for a given k, t_{in} can be written as

$$t_{in} = (\overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,k)) + x\overline{\mathcal{T}}(n,m) + \overline{\mathcal{T}}(n,m,k') \tag{8.10}$$

The value for $\bar{t}_{in}$ can be found by averaging over all the possible k values which is given by Eq. (8.9). ∎

Eq. (8.9) is bounded by

$$\begin{aligned}(1+x)\overline{\mathcal{T}}(n,m) &- \overline{\mathcal{T}}(n,m,m) + \overline{\mathcal{T}}(n,m,1) \leq \overline{t_{in}} \\ &\leq (1+x)\overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,1) + \overline{\mathcal{T}}(n,m,m)\end{aligned} \tag{8.11}$$

In Fig. 8.6 the average busy period t_{in} measure in frames is plotted.

Theorem 20: *For $m + s$ systems the average interval between two busy periods for any given system given that $m \leq s$ is bounded by*

$$\overline{\mathcal{T}}(n,m,1) + 1 \leq \overline{t_{out}} \leq 2\overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,1) + 1 \tag{8.12}$$

where $t_{out}^{UB} = 2\overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,1) + 1$ s the upper bound for the average interval between busy periods.

Proof: Assume that after the kth successful request/echo exchange within a collision-resolution round the given system looses the channel. The given system must wait until the end of the collision-resolution round before it can make a request, i.e., it must wait $\overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,k)$ frames. In the next collision-resolution round the given system will

acquire a channel, where k' can range from 1 to m. An extra frame must be added since once a successful request/echo exchange has taken the given system must wait until the end of the frame before sending its data using the channel. Therefore,

$$t_{out} = \overline{\mathcal{T}}(n,m) - \overline{\mathcal{T}}(n,m,k) + \overline{\mathcal{T}}(n,m,k') + 1 \tag{8.13}$$

The lower bound can be found by setting $k = m$ and $k' = 1$. Respectively, the upper bound can be found if $k = 1$ and $k' = m$ are set in the above equation. Therefore, t_{out} is bounded according to Eq. (8.12). $\blacksquare$

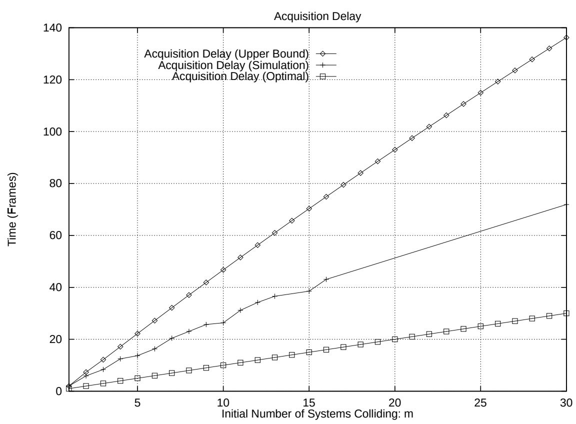

Figure 8.5: Total acquisition delay measured in frames as a function of m initial collisions.

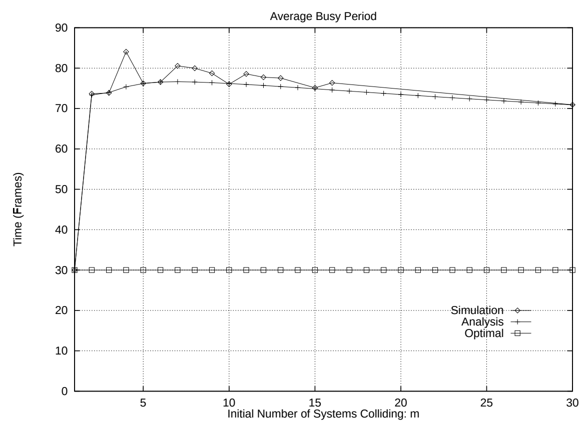

Figure 8.6: Average busy period measured in frames as a function of m initial collisions.

In Fig. 8.5 the acquisition delay t_{out} measure in frames is plotted. Given t_{in} in Theorem 19, t_{out}^{UB} in Theorem 20 for $m + s$ systems in a network and $m \leq s$, the throughput for any given system is bounded by

$$\mathcal{S} \leq \frac{\bar{t}_{in}}{\bar{t}_{in} + t_{out}^{LB}} \tag{8.14}$$

Fig. 8.4 shows simulation results and the bounds for the throughput.

8.4 Summary

We have proposed a specific set of access rules ("Spectrum Etiquette") for the general 59−59.05 GHz band. The proposed etiquette permits heterogeneous systems to co-exist with one another by means of transmissions over a control channel used to establish collision-free transmission schedules over the channels allocated for data transmission within the 59-64 GHz band. The etiquette consists of framing and signaling rules that allow systems with different PHY protocol layers to communicate, and a request resolution algorithm that assigns data channels to systems with a performance that is closed to optimum under any load of channel assignment request.

9. Conclusion

9.1 Contributions

This dissertation has addressed various issues regarding media access control (MAC) protocols. The performance of most common MAC protocols in use today rapidly degrades when stations retransmit unsuccessful packets that repeatedly collide. The motivation behind this dissertation has been to design and analyze stable collision-resolution protocols for fully connected and multi-hop networks that mitigate multiple access interference.

Our first contribution was to introduce a deterministic tree-splitting algorithm and determine the average number of idle, collision and success steps required to resolve all collisions. This is an important result, because previous calculations were based on the total number of steps and did not considered the three types of collision-resolution steps of ternary feedback models separately. Obtaining the different types of collision-resolution steps separately is critical in computing the average performance of protocols that implement collision resolution using small control packets.

We have compared the deterministic tree-splitting algorithm with the probabilistic tree-splitting algorithm introduced by Capetanakis [9] and showed that the average number of steps required by the probabilistic tree-splitting algorithm is larger than the average number of steps required by the deterministic version of the tree-splitting algorithm. More precisely, we have shown that if the number of nodes in the deterministic model goes to infinity, then the deterministic model converges from below to the probabilistic model, i.e. the probabilistic model upper bounds the deterministic one.

As an example of the integration of collision resolution of RTSs in a floor acquisition multiple access (FAMA) protocol, we have also described and analyzed a specific protocol CARMA. Our analysis shows that collision resolution significantly improves the performance of FAMA protocols. The main reason is that for a collision-resolution algorithm the average time required to resolve collisions of RTSs is much smaller than the time used to transmit the associated data packet trains, which are sent with no collisions due to floor acquisition.

Furthermore, we have shown that, as the arrival rate of RTSs increases, the throughput achieved by CARMA is close to the maximum throughput that any FAMA protocol can achieve. This is the case if propagation delays and control packets used to acquire the floor are much smaller than the data packet trains sent by the stations. An interesting feature of CARMA is that it is the first stable collision-resolution protocol for fully connected networks that does not require time slotting. It also provides a better throughput performance than FAMA or similar protocols..

To improve CARMA's throughput, transmission queues were added. The resulting protocol is called the Incremental Collision Resolution Multiple Access (ICRMA) protocol. ICRMA builds a collision-free transmission queue dynamically and the access to the channel is done during small contention periods. A single step of the deterministic collision-resolution algorithm is allowed in the contention period. After such step nodes that have been accepted in the transmission queue send packets without interference from others. We have shown, both analytically and through simulation, that ICRMA achieves a maximum throughput which is within 5% of the maximum throughput achievable by the ideal protocol. ICRMA is the first transmission queue protocol with collision resolution that dynamically changes the size of the transmission queue as load increases. ICRMA behaves like TDMA under heavy load.

To take advantage of the availability of multiple frequency channels or transmission codes, a multi-channel collision-resolution protocol was presented. More precisely, we have proposed and analyzed a new stable receiver-initiated, multi-hop, multichannel, multiple access protocol with collision resolution, called CARMA-MC. CARMA-MC is a receiver initiated protocol, which dynamically divides the channel into cycles of variable length where each cycle consists of a receiving period and a transmission period. During the receiving period, stations with one or more packets to send compete for their right to acquire the floor using a deterministic tree-splitting algorithm. Each receiving period consists of collision resolution steps. A single round of collision resolution (i.e., a success, an idle or a collision of control packets) is allowed in each contention step. The receiving period is

initiated by the receiver and takes place in the channel assigned to the receiver station. CARMA-MC provides delay guarantees as well as a lower bound on the utilization of the channel.

Finally, a prototype system was designed and analyzed for contention based systems with different PHY layers. We have proposed a specific set of access rules ("Spectrum Etiquette") for the general $59 - 59.05$ GHz band. The proposed etiquette permits heterogeneous systems to co-exist with one another by means of transmissions over a control channel used to establish collision-free transmission schedules over the channels allocated for data transmission within the 59-64 GHz band. The etiquette consists of framing and signaling rules that allow systems with different PHY protocol layers to communicate, and a request resolution algorithm that assigns data channels to systems with a performance that is closed to optimum under any load of channel assignment request. The spectrum etiquette presented in this dissertation is unique in that it is the first reported protocol based on collision resolution that works with heterogeneous physical layers.

9.2 Future Work

There are several interesting directions for future work, some are extensions of our work while others are motivated by the more general problem of mobility and information access over wireless technologies as well as by the advances in radio devices.

Our immediate future work will be to extend the results of CARMA-MC to protocols that also support transmission queues. It is also possible to extend the results obtained in CARMA-MC by using one channel for the contentions of RTSs and by separating the contention phase from the transmission phase. It would be also interesting to implement a CARMA-MC type of protocol on a hopping packet radio network.

The spectrum etiquette protocol can be improved to provide a mechanism to resolve the synchronization problem. An extension of the protocol is required to solve the hidden terminal problem.

Future research areas will be closely correlated with the development of new radio devices with the ability to receive and transmit concurrently on multiple channels. The CARMA family of algorithms can be extended to these types of radio devices.

References

[1] N. Abramson, "The ALOHA System- Another Alternative for Computer Communication," *Proceedings of the Fall Joint Computer Conference*, pp. 281-85, 1970.

[2] N. Amitay, "Resource auction multiple access (RAMA): Efficient method for fast resource assignment in decentralized wireless PCS." *Electronic Letters*, April 9, 1992, V28 N8:799-801.

[3] D. Bertsekas and R. Gallager, *Data Networks*, Second Edition, Prentice-Hall, 1992.

[4] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LANs," *Proceedings of ACM SIGCOMM '94*, pp. 212–25, ACM, 1994.

[5] K. Biba, "A Hybrid Wireless MAC Protocol Supporting Asynchronous and Synchronous MSDU Delivery Services," *Tech. Rep. Paper 802.11/91-92*, IEEE 802.11 Working Group, 1992.

[6] J. Boudenant, B. Feydel and P. Rolin, "Lynx: An IEEE802.3 compatible deterministic protocol," *Proc. IEEE INFOCOM '87*, 1987.

[7] R. L. Brewster and A. M. Glass, "Throughput analysis of non-persistent and slotted non-persistent CSMA/CA protocols," *4th International Conference on Land Mobile Radio*, pp. 231–6, Institution of Electronic and Radio Engineers, 1987.

[8] R. L. Brewster and A. M. Glass, "Simulation model of sequential multichannel network and its throughput determination," *Electronic Letters*, vol. 25, no. 15, pp. 941-942, 1989.

[9] J.I. Capetanakis, "Tree algorithm for packet broadcasting channel," *IEEE Trans. Inform. Theory*, vol.IT-25, pp. 505-515, Sept. 1979.

[10] I. Chlamtac, W. R. Franta, and K. D. Levin, "BRAM: The Broadcast Recognizing Access Method," *IEEE Trans. Commun.*, COM-27:1183-89, 1979.

[11] K. C. Chua, "Performance analysis of multichannel CSMA/CD network with noisy channel," *IEEE ICC'91*, 1991.

[12] A. Colvin, "CSMA with Collision Avoidance", *Computer Commun. vol. 6, no. 5*

[13] C. L. Fullmer and J.J. Garcia-Luna-Aceves, "Solutions to Hidden Terminal Problems in Wireless Networks," *Proc. ACM SIGCOMM 97*, Cannes, France, September 14-18, 1997.

[14] C. L. Fullmer and J.J. Garcia-Luna-Aceves, "Complete Single-Channel Solutions to Hidden Terminal Problems in Wireless LANs," *Proc. IEEE ICC'97*, Montreal, Quebec, Canada, June 8-12, 1997.

[15] C. L. Fullmer and J.J. Garcia-Luna-Aceves, "FAMA-PJ: A Channel Access Protocol for Wireless LANs," *Proc. ACM Mobile Computing and Networking '95*, Nov. 14-15, 1995.

[16] C. L. Fullmer and J.J. Garcia-Luna-Aceves, "Floor Acquisition Multiple Access for Packet-Radio Networks," *Proc. ACM SIGCOMM 95*, Cambridge, MA, August 30-September 1, 1995.

[17] R. G. Gallager, "Conflict resolution in random access broadcast networks," *Proc. AFOSR Workshop Commun. Theory Appl.*, Provincetown, MA. Sept. 1978.

[18] R. Garces and J.J. Garcia-Luna-Aceves, "Floor Acquisition Multiple Access with Collision Resolution," *Proc. ACM/IEEE MobiCom 96*, Rye, New York, November 11–12, 1996.

[19] R. Garces and J.J. Garcia-Luna-Aceves, "Collision Avoidance and Resolution Multiple Access with Transmission Groups," *Proc. IEEE INFOCOM 97*, Kobe, Japan, April 7–11, 1997.

[20] R. Garces and J.J. Garcia-Luna-Aceves, "Collision Avoidance and Resolution Multiple Access: First-Success Protocols," *Proc. IEEE ICC'97*, Montreal, Quebec, Canada, June 8-12, 1997.

[21] R. Garces and J.J. Garcia-Luna-Aceves, "A Near-Optimum Channel Access Protocol Based on Incremental Collision Resolution and Distributed Transmission Queues," *Proc. IEEE INFOCOM 98*, San Francisco, California, March 29–April 2, 1998.

[22] R. Garces, J.J. Garcia-Luna-Aceves,and R. Rom, "An Access Etiquette for Very-Wide Wireless Bands," *IEEE IC3N'98*, Lafayette, Louisiana, October 12–15, 1998.

[23] R. Garces and J.J. Garcia-Luna-Aceves, "Collision Avoidance and Resolution Multiple Access (CARMA)," accepted for publication in *Cluster Computing*, 1998.

[24] R. Garces and J.J. Garcia-Luna-Aceves, "Collision Avoidance and Resolution Multiple Access with Transmission Queues," accepted for publication in *ACM Wireless Networks Journal*, 1998.

[25] J.J. Garcia-Luna-Aceves and J. Raju, "Distributed Assignment of Codes for Multihop Packet-Radio Networks," *Proc. IEEE MILCOM'97*, Monterey, California, November 2–5, 1997.

[26] D.J. Goodman, R.A. Valenzuela, K.T. Gayliard and B. Ramamurthy, "Packet Reservation Multiple Access for Local Wireless Communications" *IEEE Transactions on Communications*, August, 1989.

[27] J.F. Hayes, "An adaptive technique for local distribution," *IEEE Trans. Commun.*, vol. COM-26, no. 8, pp. 1178–1186, 1978.

[28] L. Hu, "Local throughput performance of packet radio networks with transmitting power control," *IEEE ICC'91*, 1991.

[29] Dong Geun Jeong, Chong-Ho Choi and Wha Sook Jeon "Design and Performance Evaluation of a New Medium Access Control Protocol for Local Wireless Data Communications" *IEEE/ACM Transactions on Networking*, December 1995.

[30] P. Karn, "MACA - a new channel access method for packet radio," *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pp. 134–40, ARRL, 1990.

[31] M. J . Karol and I. Chih-Lin, "A protocol for fast resource assignment in wireless PCS," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 3, pp. 727-32, IEEE, 1994.

[32] M. J. Karol, Liu Zhao, K. Y. Eng, "Distributed-queuing request update multiple access (DQRUMA) for wireless packet (ATM) networks," *Proc. IEEE ICC '95*, Seattle, WA, USA, June 18-21, 1995.

[33] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: Part I - carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. Commun.*, vol. COM-23, no. 12, pp. 1400–1416, 1975.

[34] L. Kleinrock and M. Scholl, "Packet switching in radio channels: new conflict-free multiple access schemes," *IEEE Trans. Commun.*, vol. COM-28, no. 7, pp. 1015–1029, 1980.

[35] V. C. M. Leung, "Multichannel reservation protocol for packet multiple-access communications," *Electronic Letters*, vol. 26, no. 20, pp. 1637-1638, 1990.

[36] W. F. Lo and H. T. Mouftah, "Carrier sense multiple access with collision detection for radio channels," in *IEEE 13th International Communications and Energy Conference*, pp. 244–47, IEEE, 1984.

[37] C. G. Lof, "Packet delay for CSMA and Multi-channel ALOHA multicast schemes in WLANs with fading and co-channel interference," *IEEE Int. Conference on Universal Personal Communications*, 1996.

[38] A. M. Marsan and D. Roffinella, "Multichannel local area network protocols," *IEEE JSAC*, vol. SAC-1, no. 5, pp. 885-897, 1983.

[39] A. M. Marsan and F. Neri, "A simulation study of delay in multichannel CSMA/CD protocols," *IEEE Trans. Commun.*, vol. COM-39, no. 11, 1991.

[40] J. L. Massey, "Collision resolution algorithm and random access communications," in *Multiuser Communication Systems, G. Longo Ed.*. New York: Spriger-Verlag, 1981. pp. 73-137.

[41] A. Muir and J.J. Garcia-Luna-Aceves, "Group Allocation Multiple Access with Collision Detection," *Proc. IEEE INFOCOM 97*, Kobe, Japan, April 7–11, 1997.

[42] P802.11, D3-Un-approved Draft 3: "Wireless LAN Medium Access Control (MAC) and Physical Specifications," *IEEE*, January 1996.

[43] M. B. Pursley, "The Role of Spread Spectrum in Packet Radio Networks," *Proceedings of IEEE*, (1) pp. 116-34, January 1987.

[44] L. G. Roberts, "ALOHA packet system with and without slots and capture," *Computer Communication Review*, vol. 5, no. 2, pp. 28–42, 1972.

[45] L. G. Roberts, "Dynamic allocation of satellite capacity through packets," *Computer Communication Networks*, Netherlands, 1975.

[46] R. Rom, *Local Area and Multiple Access Networks*, Computer Science Press, 1986

[47] R. Rom, and M. Sidi, "Multiple Access Protocols," *Springer-Verlag*, New York, 1990.

[48] I. Rubin, "Access control disciplines for multi-access communications channels: reservation and TDMA schemes," *IEEE Trans. on Information Theory*, vol. IT-25, no. 5, pp. 516–536, 1979.

[49] N. Shacham and P. King, "Architecture and performance of multichannel multihop packet radio network," *IEEE JSAC*, vol. SAC-5, no. 6, pp. 1013-1025, 1987.

[50] G. S. Sidhu, R. F. Andrews, and A. B. Oppenheimer, *Inside AppleTalk, Second Edition.* Addison-Wesley Publishing Company, Inc., 1990.

[51] A. S. Tannenbaum, "Computer Networks," *Prentice Hall, Inc.*, New Jersey, 1981.

[52] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part II - the hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution," *IEEE Trans. Commun.*, vol. COM-23, no. 12, pp. 1417–1433, 1975.

[53] T. Towsley, and P. O. Vales, "Announced arrival random access protocols," *IEEE Trans. Commun.*, vol. COM-35, no. 5, pp. 513-521, May 1987.

[54] B.S. Tsybakov, and V.A. Mikhailov, "Free synchronous packets access i9n a broadcast channel with feedback," *Problems of Information Transmission*, vol. 14, no. 4, pp. 259-280, Oct.-Dec. 1978.

[55] B.S. Tsybakov, and N.B. Likhanov, "Upper bound on the capacity of random multiple access system," *Problems of Information Transmission*, vol. 23, no. 3, pp. 224-236, July-Sept. 1987.

[56] C. Wu and V. O. K. Li, "Receiver-initiated busy-tone multiple access in packet radio networks," *Proc. ACM SIGCOMM'87 Workshop: Frontiers in Computer Communications Technology*, 1987.

[57] W. Xu and G. Campbell, "A distributed queuing random access protocol for a broadcast channel," *Proc. ACM SIGCOMM'93*, San Francisco, USA, 13–17 Sept. 1993.